Routing Configuration Guide for Cisco NCS 540 Series Routers, IOS XR Release 26.1.x, 26.2.x, 26.3.x

First Published: 2026-06-09

Last Modified: 2026-09-08

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

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

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

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

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

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

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

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

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

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

© 2026 Cisco Systems, Inc. All rights reserved.

CONTENTS

CHAPTER 1 **Implementing IS-IS** 1

Enable IS-IS and Configure Level 1 or Level 2 Routing 1

Single-Topology IPv6 3

Configure Single Topology for IS-IS 4

Set SPF Interval for a Single-Topology Configuration 8

Customize Routes for IS-IS 10

Maximum Paths Per Algorithm 13

Maximum Paths Per Algorithm Per Prefix 15

Set Priority for Adding Prefixes to RIB 16

IS-IS Interfaces 17

Tag IS-IS Interface Routes 18

IS-IS static neighbor 20

How IS-IS static neighbor works 21

Configure IS-IS static neighbor 21

Limit LSP Flooding 23

Control LSP Flooding for IS-IS 24

Minimum Remaining Lifetime 27

IS-IS Authentication 28

Configure Authentication for IS-IS 28

Configure Keychains for IS-IS 30

Nonstop Forwarding 31

Configure Nonstop Forwarding for IS-IS 32

ISIS NSR 34

Configuring ISIS-NSR 34

Multiprotocol Label Switching Traffic Engineering 36

Configure MPLS Traffic Engineering for IS-IS 37

MPLS TE Forwarding Adjacency 39
Tune Adjacencies for IS-IS 39
MPLS Label Distribution Protocol IGP Synchronization 41
Configuring MPLS LDP IS-IS Synchronization 42
IS-IS Overload Bit Avoidance 43
Configure IS-IS Overload Bit Avoidance 44
IS-IS Cost Fallback on IOS XR Bundle-Ether Interface 44
IS-IS Penalty for Link Delay Anomaly 46
IS-IS penalty for link loss anomaly 47
How IS-IS penalty for link loss anomaly work 48
Configure IS-IS penalty for link loss anomaly 49
Setting an SPF interval for delaying the IS-IS SPF computations 51
Setting IETF for postponing SPF calculations 51
IS-IS auto-cost reference bandwidth 53
Inter-area route propagation limits in IS-IS 53
Deployment models for inter-area route propagation limits in IS-IS 55
Restrictions for inter-area route propagation limits in IS-IS 55
Configure a hard limit threshold on propagated routes 55
References for IS-IS 59
IS-IS Functional Overview 59
Default Routes 60
Overload Bit on Router 60
Overload Bit Configuration During Multitopology Operation 60
Attached Bit on an IS-IS Instance 60
IS-IS Support for Route Tags 61
Flood Blocking on Specific Interfaces 61
Maximum LSP Lifetime and Refresh Interval 61
Mesh Group Configuration 61
Multi-Instance IS-IS 61
Label Distribution Protocol IGP Auto-configuration 61
MPLS LDP-IGP Synchronization Compatibility with LDP Graceful Restart 62
MPLS LDP-IGP Synchronization Compatibility with IGP Nonstop Forwarding 62

CHAPTER 2 **Implementing OSPF** 63

Prerequisites for Implementing OSPF 64
Enable OSPF 64
Verify OSPF Configuration and Operation 66
Stub Area 69
Not-so-Stubby Area 69
Configure Stub and Not-So-Stubby Area Types 69
Neighbors and Adjacency for OSPF 72
Configure Neighbors for Nonbroadcast Networks 72
Authentication Strategies 76
Configure Authentication at Different Hierarchical Levels for OSPF Version 2 76
Control Frequency That Same LSA Is Originated or Accepted for OSPF 79
Virtual Link and Transit Area for OSPF 81
Create Virtual Link 81
Summarize Subnetwork LSAs on OSPF ABR 86
Route Redistribution for OSPF 88
Redistribute Routes into OSPF 88
Nonstop Forwarding for OSPF Version 2 91
Configure Nonstop Forwarding Specific to Cisco for OSPF Version 2 92
OSPF Shortest Path First Throttling 94
Configure OSPF Shortest Path First Throttling 95
Graceful Restart for OSPFv3 97
Configure OSPFv3 Graceful Restart 97
Display Information About Graceful Restart 98
OSPFv2OSPF SPF Prefix Prioritization 99
Configure OSPFv2 OSPF SPF Prefix Prioritization 100
Configure OSPF as a Provider Edge to Customer Edge (PE-CE) Protocol 103
Create Multiple OSPF Instances (OSPF Process and a VRF) 106
Label Distribution Protocol IGP Auto-configuration for OSPF 107
Configure Label Distribution Protocol IGP Auto-configuration for OSPF 108
Configure LDP IGP Synchronization: OSPF 109
OSPF Authentication Message Digest Management 111
Configure Authentication Message Digest Management for OSPF 111
GTSM TTL Security Mechanism for OSPF 114
Configure Generalized TTL Security Mechanism (GTSM) for OSPF 115

OSPF Penalty for Link Delay Anomaly 117
Limiting LSA Numbers in a OSPF Link-State Database 118
Limiting the Maximum Redistributed Type-3 LSA Prefixes in OSPF 122
Autonomous System Boundary Router Isolation and Adjacency Control for LSA Overflows 124
Restrictions for ASBR Isolation and Adjacency Control for LSA Overflows 126
Configure ASBR Isolation and Adjacency Control for LSA Overflows 126
Optimization of OSPF to Reduce TCAM Usage by LPTS 127
IGP link state 128
Enhance Network Level Isolation 128
Autonomous System Boundary Router Isolation and Adjacency Control for LSA Overflows 129
References for OSPF 131
OSPF Functional Overview 131
Comparison of Cisco IOS XR Software OSPFv3 and OSPFv2 132
OSPF Hierarchical CLI and CLI Inheritance 132
OSPF Routing Components 133
Autonomous Systems 133
Areas 133
Routers 134
OSPF Process and Router ID 134
Supported OSPF Network Types 135
Route Authentication Methods for OSPF 135
Plain Text Authentication 135
MD5 Authentication 135
Key Rollover 136
OSPF FIB Download Notification 136
Designated Router (DR) for OSPF 136
Default Route for OSPF 136
Link-State Advertisement Types for OSPF Version 2 137
Link-State Advertisement Types for OSPFv3 137
Passive Interface 139
Modes of Graceful Restart Operation 139
Restart Mode 139
Helper Mode 139
Protocol Shutdown Mode 140

Load Balancing in OSPF Version 2 and OSPFv3 141
Path Computation Element for OSPFv2 141
Management Information Base (MIB) for OSPFv3 142
VRF-lite Support for OSPFv2 142
OSPFv3 Timers Update 142

CHAPTER 3 **IGP Flexible Algorithm in IP Networks** 143
Prerequisites for IP Flexible Algorithm 144
Flexible Algorithm Definition 144
Flexible Algorithm Definition Advertisement 144
IP Flexible Algorithm Prefix Advertisement 144
IP Flexible Algorithm Participation 144
Computing IP Flexible Algorithm Paths 145
IP Flexible Algorithm Forwarding 145
Flexible Algorithm Configuration 145
Associating the IP Address to Flexible Algorithm 147
Example: Configuring IS-IS IP Flexible Algorithm 147
Verifying IP Flexible Algorithm 148
Protecting Flexible Algorithm IP Prefixes 151
Example: Enabling Flexible Algorithm Protection 151
Flexible-Algorithm Redistribution in IP Networks 156
Set an Algorithm 157
Match an Algorithm 159

CHAPTER 4 **Implementing and Monitoring RIB** 163
Verify RIB Configuration Using Routing Table 163
Verify Networking and Routing Problems 164
Disable RIB Next-hop Dampening 166
Enable RCC and LCC On-demand Scan 167
Enable RCC and LCC Background Scan 168
References for RIB 169
RIB Data Structures in BGP and Other Protocols 170
RIB Administrative Distance 170
RIB Statistics 171

RIB Quarantining 171
Route and Label Consistency Checker 172

CHAPTER 5 **Implementing Routing Policy** 173
Restrictions for Implementing Routing Policy 173
Define Route Policy 174
Attach Routing Policy to BGP Neighbor 176
Modify Routing Policy Using Text Editor 177
References for Routing Policy 180
Routing Policy Language 180
Routing Policy Language Overview 180
Routing Policy Language Structure 180
Routing Policy Language Components 188
Routing Policy Language Usage 188
Policy Definitions 190
Parameterization 191
Parameterization at Attach Points 192
Global Parameterization 192
Semantics of Policy Application 193
Boolean Operator Precedence 193
Multiple Modifications of Same Attribute 193
When Attributes Are Modified 194
Default Drop Disposition 195
Control Flow 195
Policy Verification 196
Policy Statements 198
Remark 198
Disposition 198
Action 200
If 200
Boolean Conditions 201
apply 203
Attach Points 203
BGP Policy Attach Points 204

OSPF Policy Attach Points 224
OSPFv3 Policy Attach Points 227
IS-IS Policy Attach Points 229
Nondestructive Editing of Routing Policy 229
Attached Policy Modification 230
Nonattached Policy Modification 230
Editing Routing Policy Configuration Elements 230
Hierarchical Policy Conditions 232
Apply Condition Policies 233
Nested Wildcard Apply Policy 235
VRF Import Policy Enhancement 236
Match Aggregated Route 236
Remove Private AS in Inbound Policy 236

CHAPTER 6 **Implementing RIP** 237
Prerequisites for Implementing RIP 237
Information About Implementing RIP 237
How to Implement RIP 243
Configuration Examples for Implementing RIP 246
Additional References 248

CHAPTER 7 **Implementing Static Routes** 251
Restrictions for Implementing Static Routes 251
Configure Static Route 251
Floating Static Routes 253
Configure Floating Static Route 253
Configure Static Routes Between PE-CE Routers 254
IPv4 Multicast Static Routes 256
Configure Multicast Static Routes 257
Default VRF 258
Associate VRF with a Static Route 258
References for Static Routes 259
Static Route Functional Overview 259
Default Administrative Distance 260

Directly Connected Routes 260
Floating Static Routes 260
Fully Specified Static Routes 261
Recursive Static Routes 261

CHAPTER 8 **Implementing BFD** 263
BFD Overview 263
BFD Timers 265
Enable and Disable IPv6 Checksum Calculations for BFD on a Router 266
Configure BFD Under a Dynamic Routing Protocol or Use a Static Route 266
Enable BFD for OSPF on an Interface 267
Enable BFD over BGP 268
Enable BFD on an IPv4 Static Route 268
Enable BFD on an IPv6 Static Route 269
Clear and Display BFD Counters 270
BFD Session Types 270
BFD Singlepath Sessions 270
BFD over Bundle 270
Configure BFD over Bundle 271
Enabling BFD on a BGP Neighbor 273
Enabling BFD for OSPF on an Interface 274
Enabling BFD on a Static Route 276
Enabling BFD Sessions on Bundle Members 277
Specifying the BFD Destination Address on a Bundle 278
Configuring the Minimum Thresholds for Maintaining an Active Bundle 278
Configuring BFD Packet Transmission Intervals and Failure Detection Times on a Bundle 280
Configuring BFD over Bundle per Member Mode 281
Configure BFD over Bundles IETF Mode Support on a Per Bundle Basis 282
BoB Configuration for IPv4 and IPv6 283
BFD Hardware Offload Support for IPv6 284
BFD over Bundle with IPv4 Unnumbered Interfaces 286
BFD Multipath Sessions 287
BFD over BVI 288
IPv4 Multihop BFD 289

Configure IPv4 Multihop BFD 289
Configure Multihop BFD on IPv4 Non-default VRFs 289
Multihop BFD over BVI 291
Seamless Bidirectional Forwarding Detection 294
Usage Guidelines and Limitations for S-BFD 296
Configure the S-BFD Reflector 296
Coexistence Between BFD over Bundle and BFD over Logical Bundle 298
BFD CPU Offload Support for IPv6 304
Limitations for BFD CPU offload support for IPv6 304
Configure BFD CPU Offload Support for IPv6 305
BFD Object Tracking 306
Configuring BFD Object Tracking: 306
BFD Dampening 307

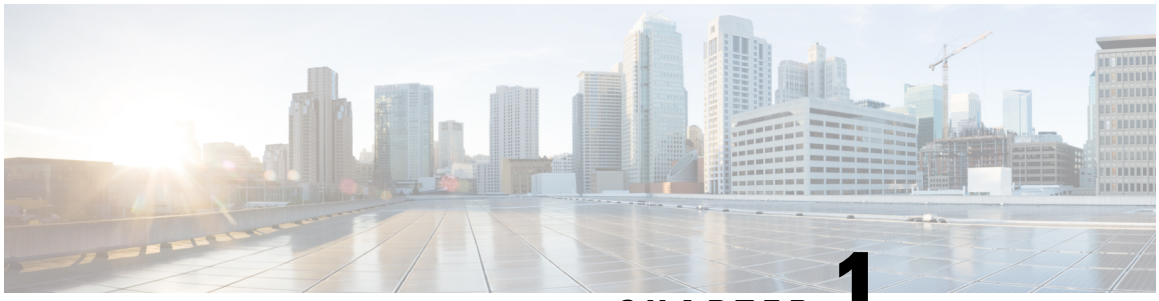
CHAPTER 9 **ECMP vs. UCMP Load Balancing** 309
UCMP Minimum Integer Ratio 309
Configuring IS-IS With Weight 310
Configuring IS-IS With Metric 311
Configuring BGP With Weights 312
Configuring TE Tunnel With Weights 313
Policy-Based Tunnel Selection 314
Interior Gateway Protocol (IGP) Destination-based Load Balancing (DLB) 326
Restrictions for IGP DLB 327
Configuring IGP DLB 327

CHAPTER 10 **Implementing Fast Reroute Loop-Free Alternate** 329
Prerequisites for Fast Reroute with Loop-Free Alternate 329
Restrictions for Fast Reroute with Loop-Free Alternate 329
IS-IS and FRR 330
Repair Paths 330
LFA Overview 330
LFA Calculation 331
Interaction Between RIB and Routing Protocols 331
Fast Reroute with Remote Loop-Free Alternate 332

Configuration 333
Running Configuration 334
Verification 335

CHAPTER 11 **Implementing EIGRP 337**
Restrictions for Implementing EIGRP 337
Information About Implementing EIGRP 337
EIGRP Functional Overview 338
EIGRP v4/v6 Authentication Using Keychain 338
Enable EIGRP Routing 339
Monitor EIGRP Routing 339
Configure Route Summarization for an EIGRP Process 340
Configure Stub Routing for an EIGRP Process 341
Configure EIGRP as a PE-CE Protocol 341
Unicast Neighbors 342
Remote Neighbor Session Policy 343
Understanding Neighbor Terms 344
Remote Unicast-Listen (Point-to-Point) Neighbors 344
Restrictions for remote neighbors 345
Inheritance and precedence of the remote neighbor configurations 345
EIGRP Features 345
EIGRP Components 346
EIGRP Configuration Grouping 346
EIGRP Configuration Modes 347
EIGRP Interfaces 348
MPLS VPN Support for EIGRP Between PE and CE 348
Redistribution for an EIGRP Process 348
Redistribute Routes for EIGRP 349
Create a Route Policy and Attach it to an EIGRP Process 349
Redistribute BGP Routes into EIGRP 350
Metric Weights for EIGRP Routing 350
Mismatched K Values 351
Goodbye Message 351
Percentage of Link Bandwidth Used for EIGRP Packets 352

Floating Summary Routes for an EIGRP Process 352
Split Horizon for an EIGRP Process 354
Adjustment of Hello Interval and Hold Time for an EIGRP Process 354
Stub Routing for an EIGRP Process 355
Route Policy Options for an EIGRP Process 356
Default-Accept-In 357
EIGRP Layer 3 VPN PE-CE Site-of-Origin 357
Router Interoperation with the Site-of-Origin Extended Community 357
Route Manipulation using SoO match condition 358
EIGRP Wide Metric Computation 359
EIGRP Multi-Instance 360



CHAPTER 1

Implementing IS-IS

Integrated Intermediate System-to-Intermediate System (IS-IS), Internet Protocol Version 4 (IPv4), is a standards-based Interior Gateway Protocol (IGP). The Cisco software implements the IP routing capabilities described in International Organization for Standardization (ISO)/International Engineering Consortium (IEC) 10589 and RFC 1195, and adds the standard extensions for single topology and multitopology IS-IS for IP Version 6 (IPv6).

This module describes how to implement IS-IS (IPv4 and IPv6) on your Cisco IOS XR network.

- Enable IS-IS and Configure Level 1 or Level 2 Routing, on page 1
- Single-Topology IPv6, on page 3
- Customize Routes for IS-IS, on page 10
- Set Priority for Adding Prefixes to RIB, on page 16
- IS-IS Interfaces, on page 17
- IS-IS static neighbor, on page 20
- Limit LSP Flooding, on page 23
- IS-IS Authentication, on page 28
- Nonstop Forwarding, on page 31
- ISIS NSR, on page 34
- Multiprotocol Label Switching Traffic Engineering, on page 36
- IS-IS Overload Bit Avoidance, on page 43
- IS-IS Cost Fallback on IOS XR Bundle-Ether Interface, on page 44
- IS-IS Penalty for Link Delay Anomaly, on page 46
- IS-IS penalty for link loss anomaly, on page 47
- Setting an SPF interval for delaying the IS-IS SPF computations, on page 51
- IS-IS auto-cost reference bandwidth, on page 53
- Inter-area route propagation limits in IS-IS, on page 53
- References for IS-IS, on page 59

Enable IS-IS and Configure 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.

Note Users can configure the **no max-metric** command only with levels 1 or 2, that is, **no max-metric level {1|2}** in order to view the result in the output of the **show configuration** command. Else, the maximum metric configuration is not displayed in the output. This behavior is observed before committing the configuration to the router.

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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

- By default, all IS-IS instances are automatically Level 1 and Level 2. You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.

Step 3 **net** *network-entity-title*

Example:

```
RP/0/RP0/CPU0:router(config-isis)# net 47.0004.004d.0001.0001.0c11.1110.00
```

Configures network entity titles (NETs) for the routing instance.

- Specify a NET for each routing instance if you are configuring multi-instance IS-IS.
- This example configures a router with area ID 47.0004.004d.0001 and system ID 0001.0c11.1110.00.
- To specify more than one area address, specify additional NETs. Although the area address portion of the NET differs, the systemID portion of the NET must match exactly for all of the configured items.

Step 4 **is-type** { **level-1** | **level-1-2** | **level-2-only** }

Example:

```
RP/0/RP0/CPU0:router(config-isis)# is-type level-2-only
```

(Optional) Configures the system type (area or backbone router).

- By default, every IS-IS instance acts as a **level-1-2** router.
- The **level-1** keyword configures the software to perform Level 1 (intra-area) routing only. Only Level 1 adjacencies are established. The software learns about destinations inside its area only. Any packets containing destinations outside the area are sent to the nearest **level-1-2** router in the area.
- The **level-2-only** keyword configures the software to perform Level 2 (backbone) routing only, and the router establishes only Level 2 adjacencies, either with other Level 2-only routers or with **level-1-2** routers.
- The **level-1-2** keyword configures the software to perform both Level 1 and Level 2 routing. Both Level 1 and Level 2 adjacencies are established. The router acts as a border router between the Level 2 backbone and its Level 1 area.

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

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

end —Prompts user to take one of these actions:

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

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

Example:

```
RP/0/RP0/CPU0:router# show isis protocol
```

(Optional) Displays summary information about the IS-IS instance.

Single-Topology IPv6

Single-topology IPv6 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.

Configure Single Topology for IS-IS

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

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

Before you begin

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

Two exceptions to these instructions exist:

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

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

Enters interface configuration mode.

Step 3 Do one of the following:

- **ipv4 address** *address mask*
- **ipv6 address** *ipv6-prefix / prefix-length* [**eui-64**]

- **ipv6 address** *ipv6-address* { */ prefix-length* | *link-local* }
- **ipv6 enable**

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.0.1.3 255.255.255.0
```

or

```
RP/0/RP0/CPU0:router(config-if)# ipv6 address 3ffe:1234:c18:1::/64 eui-64
RP/0/RP0/CPU0:router(config-if)# ipv6 address FE80::260:3EFF:FE11:6770 link-local
RP/0/RP0/CPU0:router(config-if)# ipv6 enable
```

or

Defines the IPv4 address for the interface. An IP address is required on all interfaces in an area enabled for IS-IS if any one interface is configured for IS-IS routing.

or

Specifies an IPv6 network assigned to the interface and enables IPv6 processing on the interface with the **eui-64** keyword.

or

Specifies an IPv6 address assigned to the interface and enables IPv6 processing on the interface with the **link-local** keyword.

or

Automatically configures an IPv6 link-local address on the interface while also enabling the interface for IPv6 processing.

- The link-local address can be used only to communicate with nodes on the same link.
- Specifying the **ipv6 address** *ipv6-prefix* / *prefix-length* interface configuration command without the **eui-64** keyword configures site-local and global IPv6 addresses.
- Specifying the **ipv6 address** *ipv6-prefix* / *prefix-length* command with the **eui-64** keyword configures site-local and global IPv6 addresses with an interface ID in the low-order 64 bits of the IPv6 address. Only the 64-bit network prefix for the address needs to be specified; the last 64 bits are automatically computed from the interface ID.
- Specifying the **ipv6 address** command with the **link-local** keyword configures a link-local address on the interface that is used instead of the link-local address that is automatically configured when IPv6 is enabled on the interface.

Step 4 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-if)# exit
```

Exits interface configuration mode, and returns the router to mode.

Step 5 **router isis** *instance-id*

Example:

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

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

- By default, all IS-IS instances are Level 1 and Level 2. You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.

Step 6 **net** *network-entity-title*

Example:

```
RP/0/RP0/CPU0:router(config-isis)# net 47.0004.004d.0001.0001.0c11.1110.00
```

Configures NETs for the routing instance.

- Specify a NET for each routing instance if you are configuring multi-instance IS-IS. You can specify a name for a NET and for an address.
- This example configures a router with area ID 47.0004.004d.0001 and system ID 0001.0c11.1110.00.
- To specify more than one area address, specify additional NETs. Although the area address portion of the NET differs, the system ID portion of the NET must match exactly for all of the configured items.

Step 7 **address-family ipv6** [**unicast**]

Example:

```
RP/0/RP0/CPU0:router(config-isis)# address-family ipv6 unicast
```

Specifies the IPv6 address family and enters router address family configuration mode.

- This example specifies the unicast IPv6 address family.

Step 8 **single-topology**

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# single-topology
```

(Optional) Configures the link topology for IPv4 when IPv6 is configured.

- The **single-topology** command is valid only in IPv6 submode. The command instructs IPv6 to use the single topology rather than the default configuration of a separate topology in the multitopology mode.

Step 9 exit

Example:

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

Exits router address family configuration mode, and returns the router to router configuration mode.

Step 10 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-isis)# interface HundredGigE 0/1/0/3
```

Enters interface configuration mode.

Step 11 **circuit-type** { **level-1** | **level-1-2** | **level-2-only** }

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)# circuit-type level-1-2
```

(Optional) Configures the type of adjacency.

- The default circuit type is the configured system type (configured through the **is-type** command).
- Typically, the circuit type must be configured when the router is configured as only **level-1-2** and you want to constrain an interface to form only **level-1** or **level-2-only** adjacencies.

Step 12 **address-family** { **ipv4** | **ipv6** } [**unicast**]

Example:

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

Specifies the IPv4 or IPv6 address family, and enters interface address family configuration mode.

- This example specifies the unicast IPv4 address family on the interface.

Step 13 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

Step 14 **show isis** [**instance** *instance-id*] **interface** [*type interface-path-id*] [**detail**] [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router# show isis interface HundredGigE 0/0/1/0
```

(Optional) Displays information about the IS-IS interface.

Step 15 **show isis** [**instance** *instance-id*] **topology** [**systemid** *system-id*] [**level** { **1** | **2** }] [**summary**]

Example:

```
RP/0/RP0/CPU0:router# show isis topology
```

(Optional) Displays a list of connected routers in all areas.

Configuring Single-Topology IS-IS for IPv6: Example

The following example shows single-topology mode being enabled. An IS-IS instance is created, the NET is defined, IPv6 is configured along with IPv4 on an interface, and IPv4 link topology is used for IPv6. This configuration allows POS interface 0/3/0/0 to form adjacencies for both IPv4 and IPv6 addresses.

```
router isis isp
 net 49.0000.0000.0001.00
 address-family ipv6 unicast
  single-topology
 interface POS0/3/0/0
  address-family ipv4 unicast
  !
  address-family ipv6 unicast
  !
  exit
!
interface POS0/3/0/0
 ipv4 address 10.0.1.3 255.255.255.0
 ipv6 address 2001::1/64
```

Set SPF Interval for a Single-Topology 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
Router(config)# router isis isp
```

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

- You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.

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

Example:

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

Specifies the IPv4or IPv6 address family, and enters router address family configuration mode.

Step 4 **spf-interval** {[**initial-wait** *initial* | **secondary-wait** *secondary* | **maximum-wait** *maximum*] ...} [**level** { **1** | **2** }]

Example:

```
Router(config-isis-af)# spf-interval maximum-wait 5000 initial-wait 50 secondary-wait 200
```

(Optional) Controls the minimum time between successive SPF calculations.

- This value imposes a delay in the SPF computation after an event trigger and enforces a minimum elapsed time between SPF runs.
- If this value is configured too low, the router can lose too many CPU resources when the network is unstable.
- Configuring the value too high delays changes in the network topology that result in lost packets.
- The SPF interval does not apply to the running of the ISPF because that algorithm runs immediately on receiving a changed LSP.

Step 5 **ispf** [**level** { **1** | **2** }]

Example:

```
Router(config-isis-af)# ispf
```

(Optional) Configures incremental IS-IS ISPF to calculate network topology.

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

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

end —Prompts user to take one of these actions:

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

Step 7 **show isis** [**instance** *instance-id*] [[**ipv4** | **ipv6** | **afi-all**] [**unicast** | **safi-all**]] **spf-log** [**level** { **1** | **2** }] [**fspf** | **prc** | **nhc**] [**detail** | **verbose**] [**last** *number* | **first** *number*]

Example:

```
Router# show isis instance 1 ipv4 spf-log
```

(Optional) Displays how often and why the router has run a full SPF calculation.

Customize 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

- By default, all IS-IS instances are automatically Level 1 and Level 2. You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.

Step 3 **set-overload-bit** [**on-startup** { *delay* | **wait-for-bgp** }] [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis)# set-overload-bit
```

(Optional) Sets the overload bit.

Note
The configured overload bit behavior does not apply to NSF restarts because the NSF restart does not set the overload bit during restart.

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

Example:

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

Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.

Step 5 **default-information originate** [**route-policy** *route-policy-name*]

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# default-information originate
```

(Optional) Injects a default IPv4 or IPv6 route into an IS-IS routing domain.

- The **route-policy** keyword and *route-policy-name* argument specify the conditions under which the IPv4 or IPv6 default route is advertised.
- If the **route-policy** keyword is omitted, then the IPv4 or IPv6 default route is unconditionally advertised at Level 2.

Step 6 **distribute-list** { {**prefix-list** *prefix-list-name* | **route-policy** *route-policy-name*} } **in**

Example:

```
RP/0/RP0/CPU0:router(config-isis)# distribute-list { {prefix-list | prefix-list-name} |
{route-policy | route-policy-name} } in
```

(Optional) Filters the routes that Intermediate System-to-Intermediate System (IS-IS) installs in the Routing Information Base (RIB).

Warning
When **distribute-list in** command is configured, some routes that IS-IS computes are not installed in the forwarding plane of the local router, but other IS-IS routers will not be aware of this. This introduces a difference between the forwarding state computed by other IS-IS routers and the actual forwarding state on this router. In some cases, this could lead to traffic being dropped or looped. Hence, be careful about when to use this command.

Step 7 **redistribute isis** *instance* [**level-1** | **level-2** | **level-1-2**] [**metric** *metric*] [**metric-type** { **internal** | **external** }] [**policy** *policy-name*]

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# redistribute isis 2 level-1
```

(Optional) Redistributes routes from one IS-IS instance into another instance.

- In this example, an IS-IS instance redistributes Level 1 routes from another IS-IS instance.

Step 8 Do one of the following:

- **summary-prefix** *address* / *prefix-length* [**level** { **1** | **2** }]
- **summary-prefix** *ipv6-prefix* / *prefix-length* [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# summary-prefix 10.1.0.0/16 level 1
```

or

```
RP/0/RP0/CPU0:router(config-isis-af)# summary-prefix 3003:xxxx::/24 level 1
```

(Optional) Allows a Level 1-2 router to summarize Level 1 IPv4 and IPv6 prefixes at Level 2, instead of advertising the Level 1 prefixes directly when the router advertises the summary.

- This example specifies an IPv4 address and mask.

or

- This example specifies an IPv6 prefix, and the command must be in the form documented in RFC 2373 in which the address is specified in hexadecimal using 16-bit values between colons.

- 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 9 **maximum-paths** *route-number*

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# maximum-paths 16
```

(Optional) Configures the maximum number of parallel paths allowed in a routing table.

Step 10 **distance** *weight* [*address / prefix-length* [*route-list-name*]]

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# distance 90
```

(Optional) Defines the administrative distance assigned to routes discovered by the IS-IS protocol.

- A different administrative distance may be applied for IPv4 and IPv6.

Step 11 **set-attached-bit**

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# set-attached-bit
```

(Optional) Configures an IS-IS instance with an attached bit in the Level 1 LSP.

Step 12 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

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 HundredGigE 0/0/1/0
   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 HundredGigE 0/0/1/0
   address-family ipv4 unicast
```

Maximum Paths Per Algorithm

Table 1: Feature History Table

Feature Name	Release	Description
Maximum Paths Per Algorithm	Release 7.8.1	This feature introduces the new algorithm 0 command. These updates enable individual granularity for regular SPF algorithms.

A new **algorithm 0** command is introduced.

The **algorithm 0** command includes the **address-family** <*ipv4*/*ipv6*> **unicast** subcommand, and a new **maximum-paths** <*maximum-paths*> subcommand. The **maximum-paths** under **algorithm 0** configuration block applies to the standard Shortest Path First algorithm of the IS-IS instance.

Note For information on IS-IS Flex Algo Maximum Paths, refer to the "Enabling Segment Routing Flexible Algorithm" chapter in the *Segment Routing Configuration Guide for Cisco NCS 540 Series Routers*.

The new subcommands allow for maximum number of Equal-Cost Multi-path (ECMP) to be set for individual algorithms. The value that is configured on a per-algo per address-family basis overrides any value that is configured under the IS-IS global address-family submode.

Usage Guidelines and Limitations

- The maximum-paths per algorithm takes precedence over maximum-paths per address-family.
- The maximum paths effective for each SPF algorithm are as follows:
  - For algorithm 0/Standard SPF:

- IPv4: 1
- IPv6: 2

Configuration Example – Max Path

This example shows how you can set the per-algo maximum path:

```
Router(config)# router isis isp
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# maximum-paths 12
Router(config-isis-af)# exit
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# maximum-paths 8
Router(config-isis-af)# exit

Router(config-isis)# algorithm 0
Router(config-isis-std-algo)# address-family ipv4 unicast
Router(config-isis-std-algo-af)# maximum-paths 1
Router(config-isis-std-algo)# exit
Router(config-isis-std-algo)# address-family ipv6 unicast
Router(config-isis-std-algo-af)# maximum-paths 2
Router(config-isis-std-algo-af)# exit
```

Maximum Paths Per Algorithm Per Prefix

Table 2: Feature History Table

Feature Name	Release	Description
Maximum Paths Per Flexible Algorithm Per Prefix	Release 7.11.1	Previously, you could configure a maximum number of Equal-Cost Multi-path (ECMP) to be set for SPF algo 0. This feature provides additional granularity to the IS-IS Maximum Paths Per-Algorithm feature by allowing you to specify a set of prefixes for SPF algo 0. Now you can achieve a balance between path diversity and computational and memory requirements by controlling the number of paths for each specific algorithm and destination prefix combination. This feature introduces these changes: **CLI** • **maximum-paths route-policy** *name* YANG Data Models: • This feature extends the native `Cisco-IOS-XR-clns-isis-cfg.yang` model See GitHub, Yang Data Models Navigator

Previously, you could set the maximum paths for a Shortest Path First (SPF) algorithm per address-family.

With this feature, you can further refine the maximum paths configuration by associating it with specific prefixes for each algorithm. The existing **maximum-paths** command is extended to include a **route-policy** qualifier to configure the maximum paths per algorithm per prefix-list.

When installing paths into the Routing Information Base (RIB), the system will check if a maximum paths value has been configured for algorithm 0 and the associated prefix. If such a configuration exists, it will be used instead of the existing address-family value to determine the number of paths to be installed.

Note Route policies that have the attribute **set maximum-paths** *number* are supported.

Note For information on IS-IS Flex Algo Maximum Paths per Prefix, refer to the "Enabling Segment Routing Flexible Algorithm" chapter in the *Segment Routing Configuration Guide for Cisco NCS 540 Series Routers*.

Usage Guidelines and Limitations

- The commands **maximum-paths** *maximum-paths* and **maximum-paths route-policy** *rpl-name* are mutually exclusive. You can configure either an unqualified number or a route-policy for any given IS-IS instance.
- The maximum-paths per algorithm per prefix takes precedence over maximum-paths per algorithm. Likewise, the maximum-paths per algorithm takes precedence over maximum-paths per address-family. This hierarchy ensures that the most specific configuration is prioritized when determining the maximum paths for a given algorithm and prefix combination.

Example: RPL - Prefix Set

Define a Prefix Set:

```
prefix-set isis-ipv4-L1
  20.1.0.101/32
end-set
```

Create a Route Policy:

```
route-policy isis-mp-if-L1
  if destination in isis-ipv4-L1 then
    set maximum-paths 2
  endif
end-policy
```

Configure Maximum Paths Per-Prefix:

```
router isis isp
 algorithm 0
  address-family ipv4 unicast
   maximum-paths route-policy isis-mp-if-L1
  !
```

Set Priority for Adding Prefixes to 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

Enables IS-IS routing for the specified routing process, and places the router in router configuration mode. In this example, the IS-IS instance is called isp.

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

Example:

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

Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.

Step 4 **metric-style wide** [**transition**] [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide level 1
```

Configures a router to generate and accept only wide-link metrics in the Level 1 area.

Step 5 **spf prefix-priority** [**level** { **1** | **2** }] { **critical** | **high** | **medium** } { *access-list-name* | **tag** *tag* }

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# spf prefix-priority high tag 3
```

Installs all routes tagged with the value 3 first.

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

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

end —Prompts user to take one of these actions:

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

IS-IS Interfaces

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

- Active—advertises connected prefixes and forms adjacencies. This is the default for interfaces.
- Passive—advertises connected prefixes but does not form adjacencies. The **passive** command is used to configure interfaces as passive. Passive interfaces should be used sparingly for important prefixes such as loopback addresses that need to be injected into the IS-IS domain. If many connected prefixes need

to be advertised then the redistribution of connected routes with the appropriate policy should be used instead.

- Suppressed—does not advertise connected prefixes but forms adjacencies. The **suppress** command is used to configure interfaces as suppressed.
- Shutdown—does not advertise connected prefixes and does not form adjacencies. The **shutdown** command is used to disable interfaces without removing the IS-IS configuration.

Tag IS-IS Interface Routes

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

Enables IS-IS routing for the specified routing process, and places the router in router configuration mode. In this example, the IS-IS instance is called isp.

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

Example:

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

Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.

Step 4 **metric-style wide** [**transition**] [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide level 1
```

Configures a router to generate and accept only wide link metrics in the Level 1 area.

Step 5 **exit**

Example:

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

Exits router address family configuration mode, and returns the router to router configuration mode.

Step 6 **interface** *type number*

Example:

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

Enters interface configuration mode.

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

Example:

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

Specifies the IPv4 or IPv6 address family, and enters address family configuration mode.

Step 8 **tag** *tag*

Example:

```
RP/0/RP0/CPU0:router(config-isis-if-af)# tag 3
```

Sets the value of the tag to associate with the advertised connected route.

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

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

end —Prompts user to take one of these actions:

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

Step 10 **show isis** [**ipv4** | **ipv6** | **afi-all**] [**unicast** | **safi-all**] **route** [**detail**]

Example:

```
RP/0/RP0/CPU0:router# show isis ipv4 route detail
```

Displays tag information. Verify that all tags are present in the RIB.

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
```

IS-IS static neighbor

IS-IS static neighbor is a network management feature that allows the advertisement of an Integrated Intermediate System-to-Intermediate System (IS-IS) link without forming an actual IS-IS adjacency.

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
IS-IS static neighbor	Release 25.2.1	IS-IS static neighbor allows the advertisement of an IS-IS link without forming an actual IS-IS adjacency. This feature is useful when a link is required in the topology for the controller, but IS-IS is not actively running on the link. This feature introduces these changes: CLI: • **router isis static-neighbor**

Benefits

IS-IS static neighbor provides the following benefits:

- Provides necessary link topology for controllers without active IS-IS adjacency.
- Ensures the advertised link is identical to regular IS-IS links.

Usage Guidelines

Static IS-IS links are advertised when the following conditions are met:

- IS-IS interface is up, including IP/IPv6 protocols and Connectionless Network Service (CLNS) routing IO
- Static IS-IS is configured, and
- Point-to-point (p2p) IS-IS is configured

How IS-IS static neighbor works

Summary

A **static-neighbor** configuration mode is introduced under **IS-IS interface** sub-mode. The commands in this new sub-mode allow you to specify the following details of the static IS-IS neighbor node.

- system-id: ISIS System-id of the static neighbor.
- remote address: Remote IPv4/IPv6 addresses.
- affinity: Affinity link to advertise.
- log advertisement changes: Enable logging static neighbor advertisement changes.

When IS-IS static neighbor is configured, protocol operation on the interface stops, preventing adjacency formation. Local IP subnets continue to be advertised alongside static neighbors.

Workflow

Table 4: Behavior of different configurations:

Configuration	Static Adjacency	Local IP Subnet
Passive	Advertised	Advertised
Suppressed	Advertised	Not Advertised
Shutdown	Not Advertised	Not Advertised

Configure IS-IS static neighbor

The purpose of this task is to configure Intermediate System-to-Intermediate System (IS-IS) static neighbor and its attributes by using the **static-neighbor** sub-mode.

Procedure

Step 1 First, gather the required details to configure IS-IS static neighbor on a router:

- IS-IS System-id of the static neighbor
- Remote IPv4/IPv6 addresses
- Affinity link to advertise
- log advertisement changes: Enable logging static neighbor advertisement changes

Step 2 Configure IS-IS static neighbor by using the **static-neighbor** command.

Example:

```
Router#config
Router(config)#router isis 1
```

```
Router(config-isis)#interface GigabitEthernet0/2/0/7
Router(config-isis-if)#static-neighbor
Router(config-isis-static-nbr)#system-id 0001.0002.0007
Router(config-isis-static-nbr)#remote ipv4 address 2.2.2.2
Router(config-isis-static-nbr)#affinity RED
Router(config-isis-static-nbr)#log-advertisement-changes
```

Step 3 View the running configuration using the **show static-neighbor** command.

Example:

```
router isis 1
 interface GigabitEthernet0/2/0/7
  static-neighbor
   system-id 0001.0002.0007
   remote ipv4 address 2.2.2.2
   affinity RED
   log-advertisement-changes
  !
 !
!
```

Step 4 Verify the output by using the **show isis static-neighbor <interface-name>** command.

Example:

Below is a sample output.

```
IS-IS 1 static neighbors:

System Id          Interface        State
0001.0002.0007     Gi0/2/0/3        Up

  Remote IPv4 Address: 2.2.2.2
  Remote IPv6 Address: 7::2
  Affinity: RED
  Level-1:
    Advertised in MT (Standard (IPv4 Unicast)): Yes
    Last Transition in MT (Standard (IPv4 Unicast)): 07:55:35.177
    Advertised in MT (IPv6 Unicast): Yes
    Last Transition in MT (IPv6 Unicast): 09:06:41.918
  Level-2:
    Advertised in MT (Standard (IPv4 Unicast)): Yes
    Last Transition in MT (Standard (IPv4 Unicast)): 07:55:35.183
    Advertised in MT (IPv6 Unicast): Yes
    Last Transition in MT (IPv6 Unicast): 09:06:41.917
```

New Syslog

```
RP/0/0/CPU0:Sep  9 05:22:27.362 PDT: isis[1009]: %ROUTING-ISIS-5-STATIC_NBR_CHANGE : ISIS
(1): Static Neighbor Advertisement 0001.0002.0007 (GigabitEthernet0/2/0/3) (L2) (MT-ID:
Standard (IPv4 Unicast)) Advertised
RP/0/0/CPU0:Sep  9 05:22:27.366 PDT: isis[1009]: %ROUTING-ISIS-5-STATIC_NBR_CHANGE : ISIS
(1): Static Neighbor Advertisement 0001.0002.0007 (GigabitEthernet0/2/0/3) (L1) (MT-ID:
Standard (IPv4 Unicast)) Advertised
RP/0/0/CPU0:Sep  9 05:22:27.402 PDT: isis[1009]: %ROUTING-ISIS-5-STATIC_NBR_CHANGE : ISIS
(1): Static Neighbor Advertisement 0001.0002.0007 (GigabitEthernet0/2/0/3) (L2) (MT-ID:
IPv6 Unicast) Advertised
```

Step 5 To display information about the IS-IS interface, use the **show isis interface** command.

Example:

```
Router#show isis interface Gi0/2/0/3
```

Below is a sample output:

```
Gi0/2/0/3             Enabled

<SNIP>

  Adjacency Formation:      Disabled (Passive due to static neighbor configuration)
  Prefix Advertisement:     Enabled
  Bandwidth:                20000000
  Total bandwidth:          20000000

  Circuit Type:             level-2-only (Configured: level-1-2)
  Media Type:               P2P
  Circuit Number:           0

  IPv4 BFD:                 Disabled
  IPv6 BFD:                 Disabled

  IPv4 Unicast Topology:    Enabled
    Adjacency Formation:    Disabled (Intf passive due to static neighbor configuration)
    Prefix Advertisement:   Running
          Policy (L1/L2):   -/-
    Metric (L1/L2):         10/10
    Metric fallback:

  IPv6 Unicast Topology:    Enabled
    Adjacency Formation:    Disabled (Intf passive due to static neighbor configuration)
    Prefix Advertisement:   Running
          Policy (L1/L2):   -/-
    Metric (L1/L2):         10/10

  IPv4 Address Family:      Enabled
    Protocol State:         Up
    Forwarding Address(es): Unknown (Intf passive due to static neighbor configuration)
    Global Prefix(es):      33.1.0.0/30 (0)
  IPv6 Address Family:      Enabled
    Protocol State:         Up
    Forwarding Address(es): Unknown (Intf passive due to static neighbor configuration)
    Global Prefix(es):      133:1::/126 (0)
```

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.

Control LSP Flooding for IS-IS

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

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

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

- You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.

Step 3 **lsp-refresh-interval** *seconds* [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis)# lsp-refresh-interval 10800
```

(Optional) Sets the time between regeneration of LSPs that contain different sequence numbers

- The refresh interval should always be set lower than the **max-lsp-lifetime** command.

Step 4 **lsp-check-interval** *seconds* [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis)# lsp-check-interval 240
```

(Optional) Configures the time between periodic checks of the entire database to validate the checksums of the LSPs in the database.

- This operation is costly in terms of CPU and so should be configured to occur infrequently.

Step 5 **lsp-gen-interval** { [**initial-wait** *initial* | **secondary-wait** *secondary* | **maximum-wait** *maximum*] **...** } [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis)# lsp-gen-interval maximum-wait 15 initial-wait 5
secondary-wait 5
```

(Optional) Reduces the rate of LSP generation during periods of instability in the network. Helps reduce the CPU load on the router and number of LSP transmissions to its IS-IS neighbors.

- During prolonged periods of network instability, repeated recalculation of LSPs can cause an increased CPU load on the local router. Further, the flooding of these recalculated LSPs to the other Intermediate Systems in the network causes increased traffic and can result in other routers having to spend more time running route calculations.

Step 6 **lsp-mtu** *bytes* [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis)# lsp-mtu 1300
```

(Optional) Sets the maximum transmission unit (MTU) size of LSPs.

Step 7 **max-lsp-lifetime** *seconds* [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis)# max-lsp-lifetime 11000
```

(Optional) Sets the initial lifetime given to an LSP originated by the router.

- This is the amount of time that the LSP persists in the database of a neighbor unless the LSP is regenerated or refreshed.

Step 8 **ignore-lsp-errors disable**

Example:

```
RP/0/RP0/CPU0:router(config-isis)# ignore-lsp-errors disable
```

(Optional) Sets the router to purge LSPs received with checksum errors.

Step 9 **interface** *type interface-path-id*

Example:

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

Enters interface configuration mode.

Step 10 **lsp-interval** *milliseconds* [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)# lsp-interval 100
```

(Optional) Configures the amount of time between each LSP sent on an interface.

Step 11 **csnp-interval** *seconds* [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)# csnp-interval 30 level 1
```

(Optional) Configures the interval at which periodic CSNP packets are sent on broadcast interfaces.

- Sending more frequent CSNPs means that adjacent routers must work harder to receive them.
- Sending less frequent CSNP means that differences in the adjacent routers may persist longer.

Step 12 **retransmit-interval** *seconds* [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)# retransmit-interval 60
```

(Optional) Configures the amount of time that the sending router waits for an acknowledgment before it considers that the LSP was not received and subsequently resends.

```
RP/0/RP0/CPU0:router(config-isis-if)# retransmit-interval 60
```

Step 13 **retransmit-throttle-interval** *milliseconds* [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)# retransmit-throttle-interval 1000
```

(Optional) Configures the amount of time between retransmissions on each LSP on a point-to-point interface.

- This time is usually greater than or equal to the **lsp-interval** command time because the reason for lost LSPs may be that a neighboring router is busy. A longer interval gives the neighbor more time to receive transmissions.

Step 14 **mesh-group** { *number* | **blocked** }

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)# mesh-group blocked
```

(Optional) Optimizes LSP flooding in NBMA networks with highly meshed, point-to-point topologies.

- This command is appropriate only for an NBMA network with highly meshed, point-to-point topologies.

Step 15 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 16 **show isis interface** [*type interface-path-id* | **level** { **1** | **2** }] [**brief**]

Example:

```
RP/0/RP0/CPU0:router# show isis interface HundredGigE 0/0/1/0 brief
```

(Optional) Displays information about the IS-IS interface.

Step 17 **show isis** [**instance** *instance-id*] **database** [**level** { **1** | **2** }] [**detail** | **summary** | **verbose**] [***** | *lsp-id*]

Example:

```
RP/0/RP0/CPU0:router# show isis database level 1
```

(Optional) Displays the IS-IS LSP database.

Step 18 **show isis** [**instance** *instance-id*] **lsp-log** [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router# show isis lsp-log
```

(Optional) Displays LSP log information.

Step 19 **show isis database-log** [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router# show isis database-log level 1
```

(Optional) Display IS-IS database log information.

Minimum Remaining Lifetime

The Minimum Remaining Lifetime feature prevents premature purging and unnecessary flooding of LSPs. If the Remaining Lifetime field gets corrupted during flooding, this corruption is undetectable. The consequences of such corruption depend on how the Remaining Lifetime value is altered. This feature resolves this problem by enabling IS-IS to reset the Remaining Lifetime value of the received LSP, to the maximum LSP lifetime. By default, the maximum LSP lifetime is configured as 1200 seconds and you can configure it to a different value using the **max-lsp-lifetime** *seconds* command. This action ensures that whatever be the value of Remaining Lifetime that is received, a system other than the originator of an LSP will never purge the LSP, until the LSP has existed in the database at least for maximum LSP lifetime.

If the remaining lifetime for the LSP reaches 0, the LSP is kept in the link state database for an additional 60 seconds. This additional lifetime is known as Zero Age Lifetime. If the corresponding router does not update the LSP even after the Zero Age Lifetime, the LSP is deleted from the link state database.

The Remaining Lifetime field is also useful in identifying a problem in the network. If the received LSP lifetime value is less than the Zero Age Lifetime, which is 60 seconds, IS-IS generates an error message indicating that it's a corrupted lifetime event. The sample error message is as follows:

```
Dec 14 15:36:45.663 : isis[1011]: RECV L2 LSP 1111.1111.1112.03-00 from 1111.1111.1112.03:
 possible corrupted lifetime 59 secs for L2 lsp 1111.1111.1112.03-00 from SNPA 02e9.4522.5326
 detected.
```

IS-IS saves the received remaining lifetime value in LSP database. The value is shown in the **show isis database** command output under the **Rcvd** field.

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).

Configure Authentication for IS-IS

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

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

- You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.

Step 3 **lsp-password** { **hmac-md5** | **text** } { **clear** | **encrypted** } *password* [**level** { **1** | **2** }] [**send-only**] [**snp send-only**]

Example:

```
RP/0/RP0/CPU0:router(config-isis)# lsp-password hmac-md5 clear password1 level 1
```

Configures the LSP authentication password.

- The **hmac-md5** keyword specifies that the password is used in HMAC-MD5 authentication.
- The **text** keyword specifies that the password uses cleartext password authentication.
- The **clear** keyword specifies that the password is unencrypted when entered.
- The **encrypted** keyword specifies that the password is encrypted using a two-way algorithm when entered.
- The **level 1** keyword sets a password for authentication in the area (in Level 1 LSPs and Level SNPs).
- The **level 2** keywords set a password for authentication in the backbone (the Level 2 area).
- The **send-only** keyword adds authentication to LSP and sequence number protocol data units (SNPs) when they are sent. It does not authenticate received LSPs or SNPs.
- The **snp send-only** keyword adds authentication to SNPs when they are sent. It does not authenticate received SNPs.

Note
To disable SNP password checking, the **snp send-only** keywords must be specified in the **lsp-password** command.

Step 4 **interface** *type interface-path-id*

Example:

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

Enters interface configuration mode.

Step 5 **hello-password** { **hmac-md5** | **text** } { **clear** | **encrypted** } *password* [**level** { **1** | **2** }] [**send-only**]

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)#hello-password text clear mypassword
```

Configures the authentication password for an IS-IS interface.

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

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

end —Prompts user to take one of these actions:

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

Configure 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

- You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.

Step 3 **l sp-password** **keychain** *keychain-name* [**level** { **1** | **2** }] [**send-only**] [**snp send-only**]

Example:

```
RP/0/RP0/CPU0:router(config-isis)# lsp-password keychain isis_a level 1
```

Configures the keychain.

Step 4 **interface** *type interface-path-id*

Example:

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

Enters interface configuration mode.

Step 5 **hello-password keychain** *keychain-name* [**level** { **1** | **2** }] [**send-only**]

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)#hello-password keychain isis_b
```

Configures the authentication password for an IS-IS interface.

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

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

end —Prompts user to take one of these actions:

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

Nonstop Forwarding

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

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

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

When the Cisco IOS XR router running IS-IS routing performs the process restarts, the router must perform two tasks to resynchronize its link-state database with that of its IS-IS neighbors. First, it must relearn the available IS-IS neighbors on the network without causing a reset of the neighbor relationship. Second, it must reacquire the contents of the link-state database for the network.

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

- IETF NSF
- Cisco NSF

If neighbor routers on a network segment are NSF-aware, meaning that they are running a software version that supports RFC5306, they assist a router configured with **nsf ietf** command that is restarting. IETF NSF

enables the neighbor routers provide adjacency and link-state information to help rebuild the routing information following a failover.

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

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

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

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

IS-IS timer guidelines for RP failover

A route processor (RP) failover can interrupt traffic for approximately 50 ms. Configure BFD with a total detection time greater than 50 ms to avoid BFD flap during RP failover. Calculate the total detection time by multiplying the BFD minimum interval by the multiplier.

Example:

- A BFD minimum interval of 15 ms with multiplier 3 produces a 45-ms detection time. Because this value is less than the observed interruption, BFD may flap.
- A BFD minimum interval of 20 ms with multiplier 3 produces a 60-ms detection time. Because this value exceeds the observed interruption, the BFD session is expected to remain up during the RP failover.

Configure Nonstop Forwarding for IS-IS

This task explains how to configure your router with NSF that allows the 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

- You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.

Step 3 **nsf** { **cisco** | **ietf** }

Example:

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

Enables NSF on the next restart.

- Enter the **cisco** keyword to run IS-IS in heterogeneous networks that might not have adjacent NSF-aware networking devices.
- Enter the **ietf** keyword to enable IS-IS in homogeneous networks where *all* adjacent networking devices support IETF draft-based restartability.

Step 4 **nsf interface-expires** *number*

Example:

```
RP/0/RP0/CPU0:router(config-isis)# nsf interface-expires 1
```

Configures the number of resends of an acknowledged NSF-restart acknowledgment.

- If the resend limit is reached during the NSF restart, the restart falls back to a cold restart.

Step 5 **nsf interface-timer** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-isis) nsf interface-timer 15
```

Configures the number of seconds to wait for each restart acknowledgment.

Step 6 **nsf lifetime** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-isis)# nsf lifetime 20
```

Configures the maximum route lifetime following an NSF restart.

- This command should be configured to the length of time required to perform a full NSF restart because it is the amount of time that the Routing Information Base (RIB) retains the routes during the restart.
- Setting this value too high results in stale routes.
- Setting this value too low could result in routes purged too soon.

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

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

end —Prompts user to take one of these actions:

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

Step 8 **show running-config** [*command*]

Example:

```
RP/0/RP0/CPU0:router# show running-config router isis isp
```

(Optional) Displays the entire contents of the currently running configuration file or a subset of that file.

- Verify that “nsf” appears in the IS-IS configuration of the NSF-aware device.
- This example shows the contents of the configuration file for the “isp” instance only.

ISIS NSR

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

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

Configuring ISIS-NSR

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

Step 3 **nsr**

Example:

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

Configures the NSR feature.

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

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

end —Prompts user to take one of these actions:

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

Step 5 **show isis nsr adjacency**

Example:

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

Displays adjacency information.

Step 6 **show isis nsr status**

Example:

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

Displays the NSR status information.

Step 7 **show isis nsr statistics**

Example:

```
RP/0/RP0/CPU0:router
router#show isis nsr statistics
IS-IS test NSR(v1a) MANDATORY STATS :
                                  V1 Active               V1 Standby         V2 Active
       V2 Standby
L1 ADJ:                                0                     0                  0
            0
L2 ADJ:                                2                     2                  0
            0
LIVE INTERFACE:                        4                     4                  0
            0
PTP INTERFACE:                         1                     1                  0
            0
LAN INTERFACE:                         2                     2                  0
            0
LOOPBACK INTERFACE:                    1                     1                  0
            0
TE Tunnel:                             1                     1                  0
            0
TE LINK:                               2                     2                  0
            0
NSR OPTIONAL STATS :
L1 LSP:                                0                     0                  0
            0
L2 LSP:                                4                     4                  0
            0
IPV4 ROUTES:                           3                     3                  0
            0
IPV6 ROUTES:                           4                     4                  0
            0
```

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

Multiprotocol Label Switching Traffic Engineering

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
MPLS TE Preference for Tunnels	Release 7.6.1	You can now configure the MPLS TE traffic for equal-cost multipath (ECMP) such that it flows only through TE tunnels. This is useful in scenarios where the hardware has resource constraints that limit the number of mixed ECMP routes. In earlier releases, IS-IS installed multiple ECMPs for a route in the Routing Information Base (RIB) through TE tunnels and physical interfaces by default. This feature introduces the following command:

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.

Configure MPLS Traffic Engineering for IS-IS

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

Before you begin

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

Note 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

- You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.

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

Example:

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

Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.

Step 4 **mpls traffic-eng level** { **1** | **2** }

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# mpls traffic-eng level 1
```

Configures a router running IS-IS to flood MPLS TE link information into the indicated IS-IS level.

Step 5 **mpls traffic-eng router-id** { *ip-address* | *interface-name interface-instance* }

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# mpls traffic-eng router-id loopback0
```

Specifies that the MPLS TE router identifier for the node is the given IP address or an IP address that is associated with the given interface.

Step 6 **mpls traffic-eng tunnel preferred**

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# mpls traffic-eng tunnel preferred
```

(optional) Prevents IS-IS from installing routes that use both MPLS TE tunnels and physical interfaces, and limits IS-IS to use only MPLS TE tunnels for ECMP.

Step 7 **metric-style wide** [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide level 1
```

Configures a router to generate and accept only wide link metrics in the Level 1 area.

Step 8 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

Step 9 **show isis** [**instance** *instance-id*] **mpls traffic-eng tunnel**

Example:

```
RP/0/RP0/CPU0:router# show isis instance isp mpls traffic-eng tunnel
```

(Optional) Displays MPLS TE tunnel information.

Step 10 **show isis** [**instance** *instance-id*] **mpls traffic-eng adjacency-log**

Example:

```
RP/0/RP0/CPU0:router# show isis instance isp mpls traffic-eng adjacency-log
```

(Optional) Displays a log of MPLS TE IS-IS adjacency changes.

Step 11 **show isis** [**instance** *instance-id*] **mpls traffic-eng advertisements**

Example:

```
RP/0/RP0/CPU0:router# show isis instance isp mpls traffic-eng advertisements
```

(Optional) Displays the latest flooded record from Cisco Multiprotocol Label Switching Traffic Engineering.

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.

Tune 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

- You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.

Step 3 **log adjacency changes**

Example:

```
RP/0/RP0/CPU0:router(config-isis)# log adjacency changes
```

Generates a log message when an IS-IS adjacency changes state (up or down).

Step 4 **interface** *type interface-path-id*

Example:

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

Enters interface configuration mode.

Step 5 **hello-padding** { **disable** | **sometimes** } [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)# hello-padding sometimes
```

Configures padding on IS-IS hello PDUs for an IS-IS interface on the router.

- Hello padding applies to only this interface and not to all interfaces.

Step 6 **hello-interval** *seconds* [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)#hello-interval 6
```

Specifies the length of time between hello packets that the software sends.

Step 7 **hello-multiplier** *multiplier* [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)# hello-multiplier 10
```

Specifies the number of IS-IS hello packets a neighbor must miss before the router should declare the adjacency as down.

- A higher value increases the networks tolerance for dropped packets, but also may increase the amount of time required to detect the failure of an adjacent router.
- Conversely, not detecting the failure of an adjacent router can result in greater packet loss.

Step 8 **h ello-password** { **hmac-md5** | **text** } { **clear** | **encrypted** } *password* [**level** { **1** | **2** }] [**send-only**]

Example:

```
RP/0/RP0/CPU0:router(config-isis-if)# hello-password text clear mypassword
```

Specifies that this system include authentication in the hello packets and requires successful authentication of the hello packet from the neighbor to establish an adjacency.

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

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

end —Prompts user to take one of these actions:

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

Step 10 **show isis** [**instance** *instance-id*] **adjacency** *t ype interface- path-id*] [**detail**] [**systemid** *system-id*]

Example:

```
RP/0/RP0/CPU0:router# show isis instance isp adjacency
```

(Optional) Displays IS-IS adjacencies.

Step 11 show isis adjacency-log

Example:

```
RP/0/RP0/CPU0:router# show isis adjacency-log
```

(Optional) Displays a log of the most recent adjacency state transitions.

Step 12 **show isis** [**instance** *instance-id*] **interface** [*type interface-path-id*] [**brief** | **detail**] [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router# show isis interface HundredGigE 0/0/1/0 brief
```

(Optional) Displays information about the IS-IS interface.

Step 13 **show isis** [**instance** *instance-id*] **neighbors** [*interface-type interface-instance*] [**summary**] [**detail**] [**systemid** *system-id*]

Example:

```
RP/0/RP0/CPU0:router# show isis neighbors summary
```

(Optional) Displays information about IS-IS neighbors.

MPLS Label Distribution Protocol IGP Synchronization

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

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

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

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

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

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

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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

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

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

- By default, all IS-IS instances are automatically Level 1 and Level 2. You can change the level of routing to be performed by a particular routing instance by using the **is-type** command.

Step 3 **interface** *type interface-path-id*

Example:

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

Enters interface configuration mode.

Step 4 **address-family ipv4 unicast**

Example:

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

Specifies the IPv4 address family and enters router address family configuration mode.

Step 5 **mpls ldp sync** [**level** { **1** | **2** }]

Example:

```
RP/0/RP0/CPU0:router(config-isis-if-af)# mpls ldp sync level 1
```

Enables MPLS LDP synchronization for the IPv4 address family under interface HundredGigE 0/0/1/0.

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

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

end —Prompts user to take one of these actions:

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

IS-IS Overload Bit Avoidance

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

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

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

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

mpls traffic-eng path-selection ignore overload

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

no mpls traffic-eng path-selection ignore overload

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

Configure 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 features:

- MPLS
- IS-IS

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **mpls traffic-eng path-selection ignore overload**

Example:

```
RP/0/RP0/CPU0:router(config)# mpls traffic-eng path-selection ignore overload
```

Activates IS-IS overload bit avoidance.

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
```

IS-IS Cost Fallback on IOS XR Bundle-Ether Interface

The IS-IS Cost Fallback on IOS XR Bundle-Ether Interface feature enables you to effectively manage the capacity of the network across the Bundle-Ether interfaces through a cost fallback mechanism. This mechanism increases the Bundle-Ether interface metric based on a active or total threshold. The threshold is a percentage of bandwidth.

Threshold = Bandwidth of the active bundle member links / the total bandwidth of all bundle links including both active and nonnative links.

The threshold based on the percentage of bandwidth makes it unnecessary to configure the changes as you upgrade the bandwidth.

Configuration Example

```
Router(config)# router isis 1
Router(config-isis)# is-type level-2-only
Router(config-isis)# net 47.0000.0000.0002.00
Router(config-isis)# log adjacency changes
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# exit
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# segment-routing mpls sr-prefer
Router(config-isis-af)# exit
Router(config-isis)# interface Bundle-Ether1
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric 6 level 1
Router(config-isis-af)# metric 21 level 2
Router(config-isis-af)# metric fallback bandwidth multiplier <1-100> threshold <1-100>
Router(config-isis-af)# exit
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# metric 30 level 1
Router(config-isis-af)# metric 60 level 2
Router(config-isis-af)# metric fallback bandwidth multiplier <factor; 20> threshold <#
bandwidth percentage; 80>
```

Running Configuration

```
router isis 1
 is-type level-2-only
 net 47.0000.0000.0002.00
 log adjacency changes
 address-family ipv4 unicast
 !
 address-family ipv6 unicast
  metric-style wide
 segment-routing mpls sr-prefer
 !
 interface Bundle-Ether1
  point-to-point
address-family ipv4 unicast
   metric 6 level 1
   metric 21 level 2
   metric fallback bandwidth multiplier <1-100> threshold <1-100>
  !
address-family ipv6 unicast
  metric 30 level 1
  metric 60 level 2
  metric fallback bandwidth multiplier <factor; 20> threshold <# bandwidth percentage; 80>
!
```

Verification

```
Router# show isis interface Bundle-Ether1
!!
  Bandwidth:               10000000
```

```
  Total bandwidth:          20000000
.
IPv4 Unicast Topology:    Enabled
    Adjacency Formation:    Running
    Prefix Advertisement:   Running
          Policy (L1/L2):   -/-
    Metric (L1/L2):         18/63
    Metric fallback:
      Bandwidth (L1/L2):    Active/Active
      Anomaly (L1/L2):      Inactive/Inactive
    Weight (L1/L2):         0/0
.
 IPv6 Unicast Topology:    Enabled
    Adjacency Formation:    Running
    Prefix Advertisement:   Running
          Policy (L1/L2):   -/-
    Metric (L1/L2):         600/1200
    Metric fallback:
      Bandwidth (L1/L2):    Active/Active
      Anomaly (L1/L2):      Inactive/Inactive
    Weight (L1/L2):         0/0
!!
```

The show output given below displays the information on IS-IS database.

```
Router# show isis database detail R2
Mon Jun  8 09:16:12.316 PDT

IS-IS 1 (Level-1) Link State Database
!!
  Metric: 600        MT (IPv6 Unicast) IPv6 10:0:13::/112
  Metric: 18         IP 10.0.13.0/24
  Total Level-1 LSP count: 1     Local Level-1 LSP count: 1
  IS-IS 1 (Level-2) Link State Database
  Metric: 1200       MT (IPv6 Unicast) IPv6 10:0:13::/112
  Metric: 63         IP 10.0.13.0/24
  Total Level-2 LSP count: 2     Local Level-2 LSP count: 1
!!
```

IS-IS Penalty for Link Delay Anomaly

Note For information on configuring the link delay anomaly threshold values, refer to Link Anomaly Detection with IGP Penalty in the Segment Routing Configuration Guide.

When you configure Link Anomaly Detection in SR-PM, PM sets an anomaly bit (A-bit). When IGP receives the A-bit, IGP can automatically increase the IGP metric of the link by a user-defined amount (IGP penalty). This updated IGP metric is advertised in the network to make this link undesirable or unusable. When the link recovers, PM resets the A-bit.

Note When node is reloaded, the default or configured IGP metric (without penalty) is advertised until a new measurement is available.

Configuration

```
RP/0/RSP0/CPU0:ios(config)# router isis 100
RP/0/RSP0/CPU0:ios(config-isis)# interface GigabitEthernet 0/1/0/1
RP/0/RSP0/CPU0:ios(config-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:ios(config-isis-if-af)# metric fallback anomaly delay increment 25
RP/0/RSP0/CPU0:ios(config-isis-if-af)# exit
RP/0/RSP0/CPU0:ios(config-isis-if)# exit
RP/0/RSP0/CPU0:ios(config-isis)# interface GigabitEthernet 0/1/0/2
RP/0/RSP0/CPU0:ios(config-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:ios(config-isis-if-af)# metric fallback anomaly delay multiplier 2
```

Running Configuration

```
router isis 100
 interface GigabitEthernet0/1/0/1
  address-family ipv4 unicast
   metric fallback anomaly delay increment 25
  !
 !
 interface GigabitEthernet0/1/0/2
  address-family ipv4 unicast
   metric fallback anomaly delay multiplier 2
  !
 !
!
```

IS-IS penalty for link loss anomaly

The IS-IS penalty for link loss anomaly is a mechanism that

- increases the routing metric of a network interface when the system detects abnormal packet loss and
- causes IS-IS to prefer alternative paths and improve overall network stability.

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
IS-IS penalty for link loss anomaly	Release 25.1.1	You can enhance network stability and routing efficiency by enabling the IS-IS routing protocol to dynamically adjust an interface's metric when the Performance Monitoring (PM) system detects a packet loss anomaly. This feature introduces these changes: CLI: • The **loss** keyword is introduced in the **metric fallback anomaly** command. YANG Data Models: • This feature extends the native `Cisco-IOS-XR-um-router-isis-cfg.yang` model See GitHub, Yang Data Models Navigator

This dynamic adjustment facilitates automated traffic engineering and ensures that IS-IS reroutes traffic away from links experiencing significant packet loss to healthier paths.

For information on configuring the synthetic loss measurement, see Synthetic Loss Measurement in the Segment Routing Configuration Guide.

How IS-IS penalty for link loss anomaly work

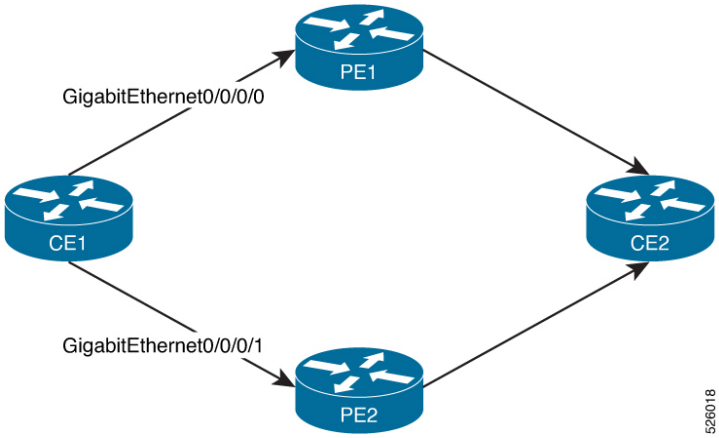
Summary

Consider a topology where Router A is connected to Router D using intermediate Routers B and C using the active Path1 and standby Path2.

- **Customer edge device**: CE1 and CE2
- **Provider edge router**: PE1 and PE2
- **CE1**: Source router
- **CE2**: Destination router
- **Path 1**: Interface GE 0/0/0/0 on CE1 connects to CE2 through PE1.
- **Path 2**: Interface GE 0/0/0/1 on CE1 connects to CE2 through PE2.
- The metric fallback anomaly loss command is configured on both GigabitEthernet0/0/0 and GigabitEthernet0/0/1 interfaces on CE1 and CE2, under the ISIS process.
- The synthetic loss measurement is configured on CE1 and CE2. For more information, see Synthetic Loss Measurement.

Workflow

Figure 1: Traffic flow under metric fallback anomaly loss



The Cisco IOS XR software takes the following steps to dynamically manage routing when the PM detects a packet loss anomaly on an interface:

1. When the measured packet loss exceeds a pre-defined upper-bound threshold in Path 1, the Cisco IOS XR software system flags this as a loss anomaly.
2. The Cisco IOS XR software communicates this loss anomaly to the ISIS routing process.
3. The metric fallback anomaly loss command, configured on the affected ISIS interface, instructs ISIS to modify the interface metric.
4. ISIS recalculates its shortest path, and the Cisco IOS XR software reroutes the traffic to Path2.
5. When the packet loss on the interface recovers and falls below the lower-bound threshold defined in the PM configuration,
   - the PM system clears the anomaly,
   - the ISIS reverts the interface metric to its original configured value, and
   - the Cisco IOS XR software reroutes the traffic to Path 1.

Configure IS-IS penalty for link loss anomaly

Perform these steps to configure and verify IS-IS penalty for link loss anomaly:

Procedure

Step 1 Execute the **interface** command to enter the ISIS interface configuration mode.

Example:

```
Router# configure terminal
Router(config)# router isis 1
Router(config-isis)# interface GigabitEthernet0/2/0/1
```

Step 2 Execute the **point-to-point** command to configure point-to-point service under the ISIS interface.

Example:

```
Router(config-isis-if)# point-to-point
```

Step 3 Execute the **address-family** and **metric fallback anomaly loss increment** commands to configure IPv4 and IPv6 address families under the ISIS interface and set the metric fallback anomaly loss increment.

Example:

```
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# metric fallback anomaly loss increment 500
Router(config-isis-if-af)# exit
Router(config-isis-if)# address-family ipv6 unicast
Router(config-isis-if-af)# metric fallback anomaly loss increment 500
Router(config-isis-if-af)# exit
Router(config-isis-if)# exit
Router(config-isis)# commit
```

Step 4 Run the **show isis database verbose** command to verify the metric fallback anomaly loss increment configuration.

Example:

```
Router# show isis database verbose r100
Fri Oct 11 10:49:17.836 PDT
IS-IS 1 (Level-2) Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime/Rcvd  ATT/P/OL
r100.00-00          * 0x0000000b   0xc725        1188 /*            0/0/0
  Metric: 10         IS-Extended r101.00
    Local Interface ID: 10, Remote Interface ID: 7
    Interface IP Address: 10.1.1.100
    Neighbor IP Address: 10.1.1.101
    Physical BW: 1000000 kbits/sec, 125000000 bytes/sec
    Link Average Delay: 704 us
    Link Min/Max Delay: 483/871 us
    Link Delay Variation: 127 us
  Metric: 10         MT (IPv6 Unicast) IS-Extended r101.00
    Local Interface ID: 10, Remote Interface ID: 7
    Interface IPv6 Address: 2001:1000::100
    Neighbor IPv6 Address: 2001:1000::101
    Physical BW: 1000000 kbits/sec, 125000000 bytes/sec
    Link Average Delay: 704 us
    Link Min/Max Delay: 483/871 us
    Link Delay Variation: 127 us
 Total Level-2 LSP count: 1     Local Level-2 LSP count: 1
```

Setting an SPF interval for delaying the IS-IS SPF computations

Table 7: Feature History Table

Feature Name	Release	Description
Setting SPF interval in IS-IS to postpone the IS-IS SPF computations	Release 7.7.1	You can now define a standard algorithm to postpone the IS-IS SPF computations by setting an SPF interval. This reduces the computational load and churn on IGP nodes when multiple temporally close network events trigger multiple SPF computations. This algorithm also reduces the probability and the duration of transient forwarding loops during native IS-IS convergence when the protocol reacts to multiple temporally close events. This feature complies with RFC 8405. This feature introduces the command.

You can set an SPF interval in IS-IS to define a standard algorithm to postpone the IS-IS SPF computations off. This reduces the computational load and churn on IGP nodes when multiple temporally close network events trigger multiple SPF computations.

This algorithm reduces the probability and the duration of transient forwarding loops during native IS-IS convergence when the protocol reacts to multiple temporally close events.

To do this, you can use the algorithm specified by RFC 8405 to temporarily postpone the IS-IS SPF computation.

This task is optional.

Setting IETF for postponing SPF calculations

Configuration

1. Enter to the Cisco IOS XR configuration mode.

   For example,

   ```
   Router# configure
   ```

2. Enable IS-IS routing for the specified routing instance and place the router in router configuration mode.

   For example,

```
Router(config)# router isis <tag>
```

3. Specify the IPv4 or IPv6 address family, and then enters router address family configuration mode.

   For example,

   ```
   Router(config-isis)# address-family {ipv4 | ipv6} unicast
   ```

4. Set the interval type (IETF) for SPF calculations.

   For example,

   ```
   Router(config-isis-af)# spf-interval ietf
   ```

5. Commit the changes.

   For example,

   ```
   Router(config-isis-af)# commit
   ```

Configuration Example

```
Router# configure
Router(config)# router isis isp
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# spf-interval ietf?
initial-wait   Initial delay before running a route calculation [50]
short-wait     Short delay before running a route calculation [200]
long-wait      Long delay before running a route calculation [5000]
learn-interval Time To Learn interval for running a route calculation [500]
holddown-interval  Holddown interval for running a route calculation [10000]
level             Set SPF interval for one level only
Router(config-isis-af)# spf-interval ietf
Router(config-isis-af)# commit
```

Verification Example

```
Router# show run router isis
router isis 1
 net 49.0001.0000.0000.0100.00
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide
  spf-interval ietf
 !
 address-family ipv6 unicast
  metric-style wide
  spf-interval ietf
 !

Router(config-isis-af)# spf-interval ietf?
initial-wait   Initial delay before running a route calculation [50]
short-wait     Short delay before running a route calculation [200]
long-wait      Long delay before running a route calculation [5000]
learn-interval Time To Learn interval for running a route calculation [500]
holddown-interval  Holddown interval for running a route calculation [10000]
level             Set SPF interval for one level only
```

The following **show** command displays the output with the new spf-interval algorithm. The output displays the actual delay taken to compute the SPF.

```
Router# show isis ipv4 spf-log last 5 detail
   IS-IS 1 Level 2 IPv4 Unicast Route Calculation Log
                   Time Total Trig.
```

```
Timestamp    Type    (ms) Nodes Count First Trigger LSP    Triggers
------------ ----- ----- ----- ----- -------------------- ------------------------
--- Wed Mar 16 2022 ---
15:31:49.763  FSPF     1     6     3         tb5-r4.00-00 LINKBAD PREFIXBAD
  Delay:                  101ms (since first trigger)
                          261177ms (since end of last calculation)
  Trigger Link:           tb5-r2.00
  Trigger Prefix:         34.1.24.0/24
  New LSP Arrivals:       0
  SR uloop:               No
  Next Wait Interval:     200ms
  RIB Batches:            1 (0 critical, 0 high, 0 medium, 1 low)
  Timings (ms):           +--Total--+
                           Real   CPU
    SPT Calculation:          1     1
    Route Update:             0     0
                          ----- -----
```

It is recommended to use the default delay values, which are listed in . These default parameters are suggetsed by RFC 8405. These should be appropriate for most networks.

However, you can configure different values if required.

For example,

```
Router# configure
Router(config)# router isis isp
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# spf-interval ietf
Router(config-isis-af)# commit
Router(config-isis-af)# spf-interval ietf short-wait 500
Router(config-isis-af)# commit
```

IS-IS auto-cost reference bandwidth

IS-IS auto-cost reference bandwidth

Table 8: Feature History Table

Feature Name	Release Name	Description
IS-IS auto-cost reference bandwidth	Release 25.4.1	Documentation in progress.

Inter-area route propagation limits in IS-IS

Inter-area route propagation limits in IS-IS are configuration parameters that

- allow operators to specify the maximum number of route prefixes that can be propagated between IS-IS levels
- provide an option to generate a warning when the configured limit is exceeded
- when enforcement is enabled, stop advertising additional prefixes once the limit is reached

- provide a warning-only mode in which all eligible prefixes are advertised, even when the limit is exceeded, and
- help protect the network from excessive routing table growth caused by misconfiguration or unexpected routing protocol events.

Table 9: Feature History Table

Feature Name	Release Name	Description
Inter-area route propagation limits in IS-IS	Release 26.3.1	This feature helps protect the network from excessive inter-area route propagation caused by misconfiguration or unexpected route growth by limiting the number of prefixes propagated between IS-IS levels. This feature, using the new **maximum-propagated-prefixes** command, allows operators to define propagation thresholds, receive early warnings, and optionally stop advertising additional prefixes when limits are exceeded. This feature introduces these changes: CLI: A new command, **maximum-propagated-prefixes**, has been introduced.

Inter-area route propagation limits in IS-IS

In IS-IS, routes learned in one level may be propagated into another level to support inter-area reachability. Prior to this enhancement, operators could control which routes were propagated by using policy, but they could not directly limit the number of propagated prefixes.

Starting with Cisco IOS XR 26.3.1, this feature adds a new command, **maximum-propagated-prefixes**, which allows operators to define a propagation threshold for inter-area routes. When the configured threshold is approached or exceeded, the router can generate warnings and, depending on configuration, either continue propagating routes or stop advertising additional prefixes. This feature helps protect the network from unexpected route growth between IS-IS levels.

When the propagated-prefix count exceeds the configured threshold, the router can either:

- generate a warning and continue advertising routes, or
- stop advertising additional propagated routes.

Benefits of inter-area route propagation limits in IS-IS

- Improved operational protection against accidental large-scale route propagation.
- Better control of IS-IS inter-area advertisements.
- Configurable warning thresholds for earlier visibility into route growth.
- Operational consistency with the existing IS-IS redistributed-prefix limiting model.

Deployment models for inter-area route propagation limits in IS-IS

The inter-area route propagation limits in IS-IS feature supports two deployment models:

- Warning-only mode (soft limit): The router generates a warning when the configured number of prefixes exceeds the soft limit, but continues advertising routes.
- Enforced mode (hard limit): The router generates a warning when a certain percentage of the hard limit is reached. IS-IS never advertises more propagated prefixes than the configured hard limit. If the number of eligible prefixes exceeds the hard limit, excess prefixes are omitted from the local system's LSPs.

These deployment models enable operators to monitor or strictly control the propagation of inter-area routes, helping prevent excessive route advertisement and maintain stable network operations. Use warning-only mode for monitoring purposes, and enforced mode when strict limits must be upheld.

Restrictions for inter-area route propagation limits in IS-IS

When you configure propagated-prefix limits in IS-IS, consider:

- Use route policy to control which prefixes are eligible for propagation.
- Use **maximum-propagated-prefixes** to control how many eligible prefixes IS-IS advertises.
- This feature applies to inter-area propagated routes. It does not limit redistributed routes.
- To limit redistributed routes, use **maximum-redistributed-prefixes**.
- To control propagation from level 2 into level 1, first enable level 2 into level 1 in IS-IS.
- Level 1 limits prefixes propagated into level 1.
- Level 2 limits prefixes propagated into level 2.
- Omit the level parameter so that the configuration applies to both levels.
- Limits are evaluated independently for the applicable area and topology.

Configure a hard limit threshold on propagated routes

Uncontrolled IS-IS route propagation can result in excessive LSP growth and increased downstream routing state. By setting a hard limit and warning thresholds, you prevent the system from advertising more routes than intended, maintaining network stability and visibility.

Before you begin

- Ensure IS-IS is enabled on the router.
- Confirm the target propagation level is part of your IS-IS design.
- Have any required route policy configured.

Follow these steps to configure a hard limit and warning-only threshold on propagated routes:

Procedure

Step 1 Enter IS-IS address-family configuration mode for the target instance.

Example:

```
Router# config
Router(config)# router isis core
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# propagate level 2 into level 1 route-policy pass-all
Router(config-isis-af)# maximum-propagated-prefixes 1000 warning-percentage 50
```

Step 2 Configure the maximum propagated prefix limit for the desired level in one of the following ways.

- This example configures the hard limit for level 2.

  ```
  Router(config-isis-af)# maximum-propagated-prefixes 100 level 2
  ```

- This example changes the default early-warning percentage from 70% to the specified value.

  ```
  Router(config-isis-af)# maximum-propagated-prefixes 100 warning-percentage 50 level 2
  ```

  The router tracks the number of propagated prefixes for the configured level. When the count exceeds the configured warning threshold, the router generates a warning. When the count exceeds the configured hard limit, the router stops advertising additional propagated routes.

  A warning message is displayed when the propagated prefix count exceeds the configured hard limit.

  ```
  Router(config-isis-af)# maximum-propagated-prefixes 100 warning-percentage 50 level 2
  RP/0/1/CPU0:Aug 24 12:52:31.457 CEST: isis[1009]: %ROUTING-ISIS-4-MAX_PROPAGATED_PREFIXES
   : ISIS (eu): Topology IPv4 Unicast is exceeding the maximum number of propagated
  prefixes (100) into level 2
  ```

- This example configures the warning-only threshold that does not enforce a hard limit.

  ```
  Router(config-isis-af)# maximum-propagated-prefixes 90 warning-only level 2
  ```

  The router generates warnings when the propagated-prefix count exceeds the configured threshold, but it continues to advertise propagated routes.

  A warning message is displayed when the propagated prefix count exceeds the soft limit.

  ```
  Router(config-isis-af)# maximum-propagated-prefixes 150 warning-percentage 50 level 2
  RP/0/1/CPU0:Aug 24 12:50:03.227 CEST: isis[1009]: %ROUTING-ISIS-4-MAX_PROPAGATED_PREFIXES
   : ISIS (eu): Topology IPv4 Unicast is exceeding the soft limit of propagated prefixes
   (150) into level 2
  ```

Step 3 Run the **show isis protocol** to verify propagated-prefix limits and review the advertised-prefix section for the relevant level.

Example:

```
Router# show isis protocol
```

```
Mon Aug 24 13:27:42.394 CEST

IS-IS Router:                       eu
  System Id:                        0000.0000.0005
  Hostname:                         ios
  IS Levels:                        level-1-2
  Manual area address:              49.0002
  Routing for area address:         49.0002
  Multi-Instance Id:                0
  Running on:                       RP/0/1/CPU0
  Job Id:                           1009
  PID:                              5734

  Started:                          Mon 2026 Aug 24 12:25:41.927 CEST
    Respawn count:                  1
    Null0 ready:                    Mon 2026 Aug 24 12:25:42.294 CEST
    First adjacency up:             Mon 2026 Aug 24 12:25:46.121 CEST
    Protocol shutdown:              No
    Non-stop forwarding:            Disabled
    Most recent startup mode:       Cold Restart

  Topologies supported by IS-IS
    IPv4 Unicast
      Distance:                     115
      Rib:                          connected
      Advertising local prefixes:   Passive Interfaces Only
      Redistributing:               no protocols
      Level-1
        Metric style:               Wide/Wide (generate/accept)
        Default Interface Metric:   10
        Advertising prefixes:       24
          Connected:                1
          Propagated:               23
            Limit:                  100
            Warning when over:      50 (50%)
          Redistributed:            0
            Limit:                  10000
            Warning when over:      7000 (70%)
      Level-2
        Metric style:               Wide/Wide (generate/accept)
        Default Interface Metric:   10
        Advertising prefixes:       101
          Connected:                1
          Propagated:               100
            Limit:                  100 (exceeded)
            Warning when over:      50 (50%)
          Redistributed:            0
            Limit:                  10000
            Warning when over:      7000 (70%)
    IPv6 Unicast
      Distance:                     115
      Rib:                          connected
      Advertising local prefixes:   Passive Interfaces Only
      Redistributing:               no protocols
      Level-1
        Default Interface Metric:   10
        Advertising prefixes:       4
          Connected:                1
          Propagated:               3
            Warning when over:      1500
          Redistributed:            0
            Limit:                  10000
            Warning when over:      7000 (70%)
      Level-2
```

```
      Default Interface Metric:  10
      Advertising prefixes:      3
        Connected:               1
        Propagated:              2
          Warning when over:     1500
        Redistributed:           0
          Limit:                 10000
          Warning when over:     7000 (70%)
Overload Bit:                    not configured
Maximum Metrics:                 not configured

LSP MTU:                         1492
LSP Full:                        level-1: No, level-2: No
LSP fast flooding:               Disabled
  Max LSP transmit speed:        1000 LSPs/sec
  PSNP delay:                    500 ms
  PSNP interval:                 50 ms

SR-MPLS:
  SRLB allocated:                15000 - 15999
  SRGB (index 0) notified:       never
  SRGB (index 0):                allocated, 16000 - 23999

MFI LSD registered:              Yes
  MFI LSD last registered:       Mon 2026 Aug 24 12:25:50.515 CEST
TE connection status:            Up
  TE connection up:              Mon 2026 Aug 24 12:25:56.877 CEST
  TE connection flaps:           0
XTC connection status:           Up
  XTC connection up:             Mon 2026 Aug 24 12:25:56.903 CEST
  XTC connection flaps:          0
BFD connection status:           Up
  BFD connection up:             Mon 2026 Aug 24 12:25:43.159 CEST
  BFD connection flaps:          0
  BFD HA role change:            never
    BFD client HA role:          none
    BFD server HA role:          none

System memory state:             Normal
  Last state change:             never
IS-IS memory state:              Normal
  Current IS-IS memory usage:    25379568 bytes (1%)
  Limit (libmemdbg):             1610612400
  Last state change:             never
  Usage during process start:    733024 bytes (0%)
  Monitoring interval:           60 sec
IS-IS memory usage per item:     items    bytes
  Nodes:                         7        2856
  LSPs:                          13       5568
  Links:                         28       34048
  Prefixes:                      596      568912
  Nets:                          234      183456
  Paths:                         520      49920

Interfaces supported by IS-IS eu:
  Loopback0 is running passively (passive in configuration)
  GigabitEthernet0/0/0/25 is disabled (active in configuration)
  GigabitEthernet0/0/0/35 is running actively (active in configuration)
  GigabitEthernet0/0/0/45 is running actively (active in configuration)
  GigabitEthernet0/0/0/51 is disabled (active in configuration)
  GigabitEthernet0/0/0/54 is disabled (active in configuration)
  GigabitEthernet0/0/0/56 is running actively (active in configuration)
  GigabitEthernet0/0/0/65 is running actively (active in configuration)
```

Review the advertised prefix counts and configured limits for each level.

Step 4 Execute the **show isis route detail** command to display a route that is not advertised because the propagation limit was exceeded.

Example:

```
Router# show isis route 2.4.1.197/32 detail
Mon Aug 24 12:54:55.932 CEST

L1 2.4.1.197/32 [110/115] Label: None, medium priority
   Installed Aug 24 12:48:46.944 for 00:06:10
     via 1.1.45.4, GigabitEthernet0/0/0/45, brat, SRGB Base: 16000, Weight: 0
     src brat.00-02, 1.1.100.4
   L2 adv [110] native, propagated, over limit: not advertised
Router#
```

IS-IS generates warnings when the propagated prefix count exceeds the configured threshold, and stops advertising additional propagated routes once the hard limit is reached. If only a warning threshold is set, IS-IS continues advertising routes but generates notifications.

What to do next

Monitor route propagation levels using router logs and the **show isis protocol** command to ensure limits are effective.

References for IS-IS

This section provides additional conceptual information on IS-IS. It includes the following topics:

IS-IS Functional Overview

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

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

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

Multiple IS-IS instances can exist on the same physical interface. However, you must configure different instance-id for every instance that shares the same physical interface.

Alternatively, you can also create dot1q sub-interfaces and configure each dot1q sub-interface to different IS-IS instances.

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 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.

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.

Attached Bit on an IS-IS Instance

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

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

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

IS-IS Support for Route Tags

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

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.

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.

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.

Multi-Instance IS-IS

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.

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.

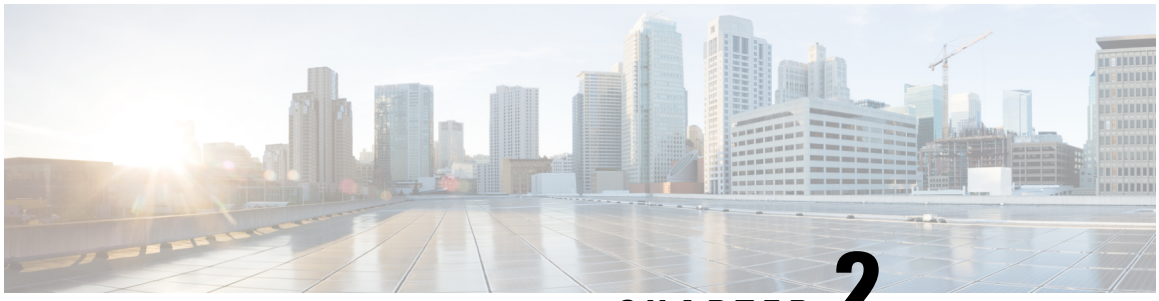
MPLS LDP-IGP Synchronization Compatibility with LDP Graceful Restart

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.



CHAPTER 2

Implementing OSPF

Open Shortest Path First (OSPF) is an Interior Gateway Protocol (IGP) developed by the OSPF working group of the Internet Engineering Task Force (IETF). Designed expressly for IP networks, OSPF supports IP subnetting and tagging of externally derived routing information. OSPF also allows packet authentication when sending and receiving packets.

OSPF Version 3 (OSPFv3) expands on OSPF Version 2, providing support for IPv6 routing prefixes.

This module describes the concepts and tasks you need to implement both versions of OSPF on your software. The term “OSPF" implies both versions of the routing protocol, unless otherwise noted.

Note GTSM TTL Security is not supported.

- Prerequisites for Implementing OSPF , on page 64
- Enable OSPF, on page 64
- Verify OSPF Configuration and Operation, on page 66
- Stub Area, on page 69
- Neighbors and Adjacency for OSPF, on page 72
- Authentication Strategies, on page 76
- Control Frequency That Same LSA Is Originated or Accepted for OSPF, on page 79
- Virtual Link and Transit Area for OSPF, on page 81
- Summarize Subnetwork LSAs on OSPF ABR, on page 86
- Route Redistribution for OSPF, on page 88
- Nonstop Forwarding for OSPF Version 2, on page 91
- OSPF Shortest Path First Throttling, on page 94
- Graceful Restart for OSPFv3, on page 97
- OSPFv2OSPF SPF Prefix Prioritization, on page 99
- Configure OSPF as a Provider Edge to Customer Edge (PE-CE) Protocol, on page 103
- Create Multiple OSPF Instances (OSPF Process and a VRF), on page 106
- Label Distribution Protocol IGP Auto-configuration for OSPF, on page 107
- OSPF Authentication Message Digest Management, on page 111
- GTSM TTL Security Mechanism for OSPF, on page 114
- OSPF Penalty for Link Delay Anomaly, on page 117
- Limiting LSA Numbers in a OSPF Link-State Database, on page 118

- Autonomous System Boundary Router Isolation and Adjacency Control for LSA Overflows, on page 124
- Optimization of OSPF to Reduce TCAM Usage by LPTS, on page 127
- IGP link state, on page 128
- Enhance Network Level Isolation, on page 128
- References for OSPF , on page 131

Prerequisites for Implementing OSPF

The following are prerequisites for implementing OSPF:

- Configuration tasks for OSPFv3 assume that you are familiar with IPv6 addressing and basic configuration. See the *Implementing Network Stack IPv4 and IPv6* in the *Cisco 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.

Enable 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 Do one of the following:

- **router ospf** *process-name*

- **router ospfv3** *process-name*

Example:

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

or

```
RP/0/RP0/CPU0:router(config)# router ospfv3 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

or

Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **router-id** { *router-id* }

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3
```

Configures a router ID for the OSPF process.

Note
We recommend using a stable IP address as the router ID.

Step 4 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# area 0
```

Enters area configuration mode and configures an area for the OSPF process.

- Backbone areas have an area ID of 0.
- Nonbackbone areas have a nonzero area ID.
- The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces for the area configured in Step 4.

Step 6 Repeat Step 5 for each interface that uses OSPF.

—

Step 7 **log adjacency changes** [**detail**] [**enable** | **disable**]

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar-if)# log adjacency changes detail
```

(Optional) Requests notification of neighbor changes.

- By default, this feature is enabled.
- The messages generated by neighbor changes are considered notifications, which are categorized as severity Level 5 in the **logging console** command. The **logging console** command controls which severity level of messages are sent to the console. By default, all severity level messages are sent.

Step 8 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

Enable OSPF: Example

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.

```
interface TenGigE 0/0/0/0
 ip address 10.1.2.1 255.255.255.0
 negotiation auto
!
interface TenGigE 0/0/0/1
 ip address 10.1.3.1 255.255.255.0
 negotiation auto
!
router ospf 1
 router-id 10.2.3.4
area 0
 interface TenGigE 0/0/0/0
!
area 1
 interface TenGigE 0/0/0/1
!
!
```

Verify OSPF Configuration and Operation

This task explains how to verify the configuration and operation of OSPF.

Procedure

Step 1 **show** { **ospf** | **ospfv3** } [*process-name*]

Example:

```
RP/0/RP0/CPU0:router# show ospf group1
```

(Optional) Displays general information about OSPF routing processes.

Step 2 **show** { **ospf** | **ospfv3** } [*process-name*] **border-routers** [*router-id*]

Example:

```
RP/0/RP0/CPU0:router# show ospf group1 border-routers
```

(Optional) Displays the internal OSPF routing table entries to an ABR and ASBR.

Step 3 **show** { **ospf** | **ospfv3** } [*process-name*] **database**

Example:

```
RP/0/RP0/CPU0:router# show ospf group2 database
```

(Optional) Displays the lists of information related to the OSPF database for a specific router.

- The various forms of this command deliver information about different OSPF LSAs.

Step 4 **show** { **ospf** | **ospfv3** } [*process-name*] [*area-id*] **flood-list interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router# show ospf 100 flood-list  TenGigE 0/0/0/0
```

(Optional) Displays a list of OSPF LSAs waiting to be flooded over an interface.

Step 5 **show** { **ospf** | **ospfv3** } [*process-name*] [**vrf** *vrf-name*] [*area-id*] **interface** [*type interface-path-id*]

Example:

```
RP/0/RP0/CPU0:router# show ospf 100 interface TenGigE 0/0/0/0
```

Example:

```
RP/0/RP0/CPU0:router# show ospf interface
Interfaces for OSPF test
TenGigE0/0/0/7 is up, line protocol is up
  Internet Address 7.7.7.3/16, Area 1
  Label stack Primary label 0 Backup label 0 SRTE label 0
  Process ID test, Router ID 7.7.7.3, Network Type BROADCAST, Cost: 1 (RSVP-TE)
  Transmit Delay is 1 sec, State DR, Priority 1, MTU 1500, MaxPktSz 1500
  Forward reference No, Unnumbered no,  Bandwidth 10000000
  Designated Router (ID) 7.7.7.3, Interface address 7.7.7.3
  Backup Designated router (ID) 7.7.7.2, Interface address 7.7.7.2
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:01:801
  Index 1/1, flood queue length 0
  Next 0(0)/0(0)
  Last flood scan length is 2, maximum is 2
```

```
  Last flood scan time is 0 msec, maximum is 0 msec
  LS Ack List: current length 0, high water mark 1
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 7.7.7.2  (Backup Designated Router)
  Suppress hello for 0 neighbor(s)
  Keychain-based authentication enabled
    Key id used is 1
    Cryptographic algorithm HMAC-SHA-256
  Multi-area interface Count is 0
```

(Optional) Displays OSPF interface information.

Step 6 **show** { **ospf** | **ospfv3** }[*process-name*] [*area-id*] **neighbor** [*t ype interface- path-id*] [*neighbor-id*] [**detail**]

Example:

```
RP/0/RP0/CPU0:router# show ospf 100 neighbor
```

(Optional) Displays OSPF neighbor information on an individual interface basis.

Step 7 **clear** { **ospf** | **ospfv3** }[*process-name*] **process**

Example:

```
RP/0/
/CPU0:router# clear ospf 100 process
```

(Optional) Resets an OSPF router process without stopping and restarting it.

Step 8 **clear**{**ospf**|**ospfv3**[*process-name*] **redistribution**

Example:

```
RP/0/RP0/CPU0:router#clear ospf 100 redistribution
```

Clears OSPF route redistribution.

Step 9 **clear**{**ospf**|**ospfv3**[*process-name*] **routes**

Example:

```
RP/0/RP0/CPU0:router#clear ospf 100 routes
```

Clears OSPF route table.

Step 10 **clear**{**ospf**|**ospfv3**[*process-name*] **vrf** [*vrf-name*|**all**] {**process** |**redistribution**|**routes**|**statistics** [**interface** *type interface-path-id*|**message-queue**|**neighbor**]}

Example:

```
RP/0/RP0/CPU0:router#clear ospf 100 vrf vrf_1 process
```

Clears OSPF route table.

Step 11 **clear** { **ospf** | **ospfv3** }[*process-name*] **statistics** [**neighbor** [*type interface-path-id*] [*ip-address*]]

Example:

```
RP/0/RP0/CPU0:router# clear ospf 100 statistics
```

(Optional) Clears the OSPF statistics of neighbor state transitions.

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.

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. Use the **capability type7 translate zero-forward-addr** command to enable the translation of Type-7 LSA with a zero forwarding address into Type-5 LSA on the NSSA ABR router. This command is supported only for OSPFv3. 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.

Configure Stub and Not-So-Stubby Area Types

This task explains how to configure the stub area and the NSSA for OSPF.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 Do one of the following:

- **router ospf** *process-name*
- **router ospfv3** *process-name*

Example:

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

or

```
RP/0/RP0/CPU0:router(config)# router ospfv3 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

or

Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **capability type7 translate zero-forward-addr**

Example:

```
RP/0/RP0/CPU0:router(config)# router ospfv3 1
RP/0/RP0/CPU0:router(config-ospfv3)# capability type7 translate zero-forward-addr
```

Enables the translation of Type-7 LSA with a zero forwarding address into Type-5 LSA on the NSSA ABR router.

Note
You can configure **capability type7 translate zero-forward-addr** command only in **router ospfv3** configuration mode. This command is not supported for the OSPF process.

Step 4 **router-id** { *router-id* }

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3
```

Configures a router ID for the OSPF process.

Note
We recommend using a stable IP address as the router ID.

Step 5 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# area 1
```

Enters area configuration mode and configures a nonbackbone area for the OSPF process.

- The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 6 Do one of the following:

- **stub** [**no-summary**]
- **nssa** [**no-redistribution**] [**default-information-originate**] [**no-summary**]

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# stub no summary
```

or

```
RP/0/RP0/CPU0:router(config-ospf-ar)# nssa no-redistribution
```

Defines the nonbackbone area as a stub area.

- Specify the **no-summary** keyword to further reduce the number of LSAs sent into a stub area. This keyword prevents the ABR from sending summary link-state advertisements (Type 3) in the stub area.

or

Defines an area as an NSSA.

Step 7 Do one of the following:

- **stub**
- **nssa**

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# stub
```

or

```
RP/0/RP0/CPU0:router(config-ospf-ar)# nssa
```

(Optional) Turns off the options configured for stub and NSSA areas.

- If you configured the stub and NSSA areas using the optional keywords (**no-summary** , **no-redistribution** , **default-information-originate** , and **no-summary**) in Step 5, you must now reissue the **stub** and **nssa** commands without the keywords—rather than using the **no** form of the command.
- For example, the **no nssa default-information-originate** form of the command changes the NSSA area into a normal area that inadvertently brings down the existing adjacencies in that area.

Step 8 **default-cost** *cost*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)#default-cost 15
```

(Optional) Specifies a cost for the default summary route sent into a stub area or an NSSA.

- Use this command only on ABRs attached to the NSSA. Do not use it on any other routers in the area.
- The default cost is 1.

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

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

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.

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

Step 10 Repeat this task on all other routers in the stub area or NSSA.

—

Configuring a Stub area: example

The following example shows that area 1 is configured as a stub area:

```
router ospfv3 1
 router-id 10.0.0.217
 area 0
  interface TenGigE 0/0/0/1
 area 1
  stub
  interface TenGigE 0/0/0/0
```

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.

Configure 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 Do one of the following:

- **router ospf** *process-name*
- **router ospfv3** *process-name*

Example:

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

or

```
RP/0/RP0/CPU0:router(config)# router ospfv3 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

or

Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **router-id** { *router-id* }

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3
```

Configures a router ID for the OSPF process.

Note
We recommend using a stable IP address as the router ID.

Step 4 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# area 0
```

Enters area configuration mode and configures an area for the OSPF process.

- The example configures a backbone area.
- The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 5 **network** { **broadcast** | **non-broadcast** }

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# network non-broadcast
```

Configures the OSPF network type to a type other than the default for a given medium.

- The example sets the network type to NBMA.

Step 6 **dead-interval** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# dead-interval 40
```

(Optional) Sets the time to wait for a hello packet from a neighbor before declaring the neighbor down.

Step 7 **hello-interval** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# hello-interval 10
```

(Optional) Specifies the interval between hello packets that OSPF sends on the interface.

Step 8 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces for the area configured in Step 4.

- In this example, the interface inherits the nonbroadcast network type and the hello and dead intervals from the areas because the values are not set at the interface level.

Step 9 Do one of the following:

- **neighbor** *ip-address* [**priority** *number*] [**poll-interval** *seconds*][**cost** *number*]
- **neighbor** *ipv6-link-local-address* [**priority** *number*] [**poll-interval** *seconds*][**cost** *number*] [**database-filter** [**all**]]

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar-if)# neighbor 10.20.20.1 priority 3 poll-interval 15
```

or

```
RP/0/RP0/CPU0:router(config-ospf-ar-if)# neighbor fe80::3203:a0ff:fe9d:f3fe
```

Configures the IPv4 address of OSPF neighbors interconnecting to nonbroadcast networks.

or

Configures the link-local IPv6 address of OSPFv3 neighbors.

- The *ipv6-link-local-address* argument must be in the form documented in RFC 2373 in which the address is specified in hexadecimal using 16-bit values between colons.
- The **priority** keyword notifies the router that this neighbor is eligible to become a DR or BDR. The priority value should match the actual priority setting on the neighbor router. The neighbor priority default value is zero.
- Neighbors with no specific cost configured assumes the cost of the interface, based on the **cost** command.
- The **database-filter** keyword filters outgoing LSAs to an OSPF neighbor. If you specify the **all** keyword, incoming and outgoing LSAs are filtered.

Step 10 Repeat Step 9 for all neighbors on the interface.

—

Step 11 exit

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar-if)# exit
```

Enters area configuration mode.

Step 12 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces for the area configured in Step 4.

- In this example, the interface inherits the nonbroadcast network type and the hello and dead intervals from the areas because the values are not set at the interface level.

Step 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**]]

Example:

```
RP/0/
/CPU0:router(config-ospf-ar)# neighbor 10.34.16.6
```

or

```
RP/0/
/CPU0:router(config-ospf-ar)# neighbor fe80::3203:a0ff:fe9d:f3f
```

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.
- Neighbors with no specific cost configured assumes the cost of the interface, based on the **cost** command.
- 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 traffic disruption or routing loops.

Step 14 Repeat Step 13 for all neighbors on the interface.

—

Step 15 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

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.

Configure Authentication at Different Hierarchical Levels for OSPF Version 2

This task explains how to configure MD5 (secure) authentication on the OSPF router process, configure one area with plain text authentication, and then apply one interface with clear text (null) authentication.

Note Authentication configured at the interface level overrides authentication configured at the area level and the router process level. If an interface does not have authentication specifically configured, the interface inherits the authentication parameter value from a higher hierarchical level.

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 OSPF Hierarchical CLI and CLI Inheritance, on page 132 for information about each type of authentication and when you should use a specific method for your network.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router ospf** *process-name*

Example:

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

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **router-id** { *router-id* }

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3
```

Configures a router ID for the OSPF process.

Step 4 **authentication** [**message-digest** | **null**]

Example:

```
RP/0/RP0/CPU0:router(config-ospf)#authentication message-digest
```

Enables MD5 authentication for the OSPF process.

- This authentication type applies to the entire router process unless overridden by a lower hierarchical level such as the area or interface.

Step 5 **message-digest-key** *key-id* **md5** { *key* | **clear** *key* | **encrypted** *key* | **LINE**}

Example:

```
RP/0/RP0/CPU0:router(config-ospf)#message-digest-key 4 md5 yourkey
```

Specifies the MD5 authentication key for the OSPF process.

- The neighbor routers must have the same key identifier.

Step 6 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# area 0
```

Enters area configuration mode and configures a backbone area for the OSPF process.

Step 7 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces to the backbone area.

- All interfaces inherit the authentication parameter values specified for the OSPF process (Step 4, Step 5, and Step 6).

Step 8 Repeat Step 7 for each interface that must communicate, using the same authentication.

—

Step 9 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# exit
```

Enters area OSPF configuration mode.

Step 10 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# area 1
```

Enters area configuration mode and configures a nonbackbone area 1 for the OSPF process.

- The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 11 **authentication** [**message-digest** | **null**]

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# authentication
```

Enables Type 1 (plain text) authentication that provides no security.

- The example specifies plain text authentication (by not specifying a keyword). Use the **authentication-key** command in interface configuration mode to specify the plain text password.

Step 12 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces to the nonbackbone area 1 specified in Step 7.

- All interfaces configured inherit the authentication parameter values configured for area 1.

Step 13 Repeat Step 12 for each interface that must communicate, using the same authentication.

—

Step 14 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces to a different authentication type.

Step 15 **authentication** [**message-digest** | **null**]

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar-if)# authentication null
```

Specifies no authentication on TenGigE 0/0/0/0, overriding the plain text authentication specified for area 1.

- By default, all of the interfaces configured in the same area inherit the same authentication parameter values of the area.

Step 16 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

Control Frequency That 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 Do one of the following:

- **router ospf** *process-name*
- **router ospfv3** *process-name*

Example:

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

or

```
RP/0/RP0/CPU0:router(config)# router ospfv3 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

or

Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **router-id** { *router-id* }

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3
```

Configures a router ID for the OSPF process.

Note
We recommend using a stable IP address as the router ID.

Step 4 Perform Step 5 or Step 6 or both to control the frequency that the same LSA is originated or accepted.

—

Step 5 **timers lsa refresh** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# timers lsa refresh 1800
```

Sets how often self-originated LSAs should be refreshed, in seconds.

- The default is 1800 seconds for both OSPF and OSPFv3.

Step 6 **timers lsa min-arrival** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# timers lsa min-arrival 2
```

Limits the frequency that new processes of any particular OSPF Version 2 LSA can be accepted during flooding.

- The default is 1 second.

Step 7 **timers lsa group-pacing** *seconds*

Example:

```
RP/0/
/CPU0:router(config-ospf)# timers lsa group-pacing 1000
```

Changes the interval at which OSPF link-state LSAs are collected into a group for flooding.

- The default is 240 seconds.

Step 8 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

Virtual Link and Transit Area for OSPF

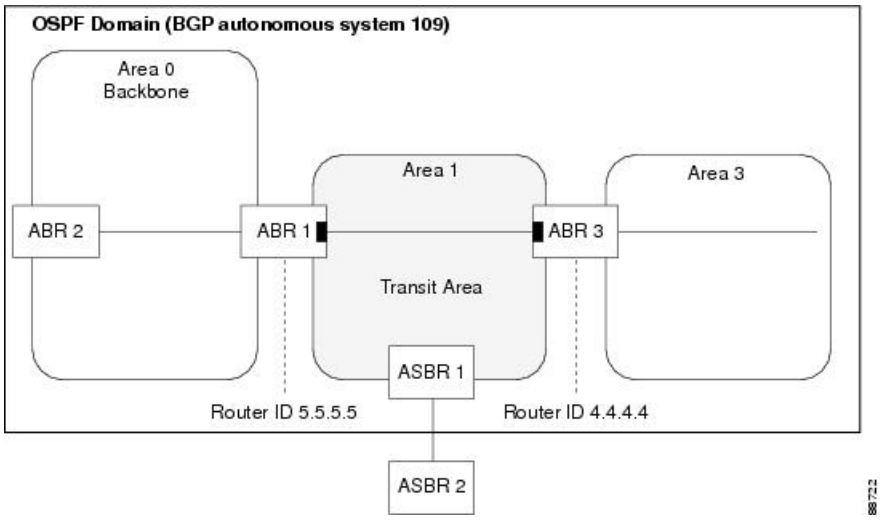
In OSPF, routing information from all areas is first summarized to the backbone area by ABRs. The same ABRs, in turn, propagate such received information to their attached areas. Such hierarchical distribution of routing information requires that all areas be connected to the backbone area (Area 0). Occasions might exist for which an area must be defined, but it cannot be physically connected to Area 0. Examples of such an occasion might be if your company makes a new acquisition that includes an OSPF area, or if Area 0 itself is partitioned.

In the case in which an area cannot be connected to Area 0, you must configure a virtual link between that area and Area 0. The two endpoints of a virtual link are ABRs, and the virtual link must be configured in both routers. The common nonbackbone area to which the two routers belong is called a transit area. A virtual link specifies the transit area and the router ID of the other virtual endpoint (the other ABR).

A virtual link cannot be configured through a stub area or NSSA.

Figure 2: Virtual Link to Area 0

This figure illustrates a virtual link from Area 3 to Area 0.



Create Virtual Link

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.

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.

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. . 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.

Procedure

Step 1 Do one of the following:

- **show ospf** [*process-name*]
- **show ospfv3** [*process-name*]

Example:

```
RP/0//CPU0:router# show ospf
```

or

```
RP/0//CPU0:router# show ospfv3
```

(Optional) Displays general information about OSPF routing processes.

- The output displays the router ID of the local router. You need this router ID to configure the other end of the link.

Step 2 **configure**

Example:

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

Enters mode.

Step 3 Do one of the following:

- **router ospf** *process-name*
- **router ospfv3** *process-name*

Example:

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

or

```
RP/0//CPU0:router(config)# router ospfv3 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

or

Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 4 **router-id** { *router-id* }

Example:

```
RP/0//CPU0:router(config-ospf)# router-id 192.168.4.3
```

Configures a router ID for the OSPF process.

Note
We recommend using a stable IPv4 address as the router ID.

Step 5 **area** *area-id*

Example:

```
RP/0//CPU0:router(config-ospf)# area 1
```

Enters area configuration mode and configures a nonbackbone area for the OSPF process.

- The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 6 **virtual-link** *router-id*

Example:

```
RRP/0//CPU0:router(config-ospf-ar)# virtual-link 10.3.4.5
```

Defines an OSPF virtual link.

- See .

Step 7 **authentication message-digest**

Example:

```
RP/0//CPU0:router(config-ospf-ar-vl)#authentication message-digest
```

Selects MD5 authentication for this virtual link.

Step 8 **message-digest-key** *key-id* **md5** { *key* | **clear** *key* | **encrypted** *key* }

Example:

```
RP/0//CPU0:router(config-ospf-ar-vl)#message-digest-key 4 md5 yourkey
```

Defines an OSPF virtual link.

- See to understand a virtual link.
- The *key-id* argument is a number in the range from 1 to 255. The *key* argument is an alphanumeric string of up to 16 characters. The routers at both ends of the virtual link must have the same key identifier and key to be able to route OSPF traffic.
- The **authentication-key** *key* command is not supported for OSPFv3.
- Once the key is encrypted it must remain encrypted.

Step 9 Repeat all of the steps in this task on the ABR that is at the other end of the virtual link. Specify the same key ID and key that you specified for the virtual link on this router.

—

Step 10 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

Step 11 Do one of the following:

- **show ospf** [*process-name*] [*area-id*] **virtual-links**
- **show ospfv3** [*process-name*] **virtual-links**

Example:

```
RP/0//CPU0:router# show ospf 1 2 virtual-links
```

or

```
RP/0//CPU0:router# show ospfv3 1 virtual-links
```

(Optional) Displays the parameters and the current state of OSPF virtual links.

ABR 1 Configuration

ABR 2 Configuration

Creating an OSPFv3 virtual link

In the following example, the **show ospfv3 virtual links** 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 TenGigE 0/1/0/10/0/0/0 Cost of using 2
    Transmit Delay is 5 sec,
    Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
      Hello due in 00:00:02
      Adjacency State FULL (Hello suppressed)
      Index 0/2/3, retransmission queue length 0, number of retransmission 1
      First 0(0)/0(0)/0(0) Next 0(0)/0(0)/0(0)
      Last retransmission scan length is 1, maximum is 1
      Last retransmission scan time is 0 msec, maximum is 0 msec

  Check for lines:
  Virtual Link OSPF_VL0 to router 10.0.0.3 is up
      Adjacency State FULL (Hello suppressed)

  State is up and Adjacency State is FULL
```

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:

```
router ospfv3 1
 router-id 10.0.0.217
 area 0
  interface TenGigE 0/2/0/10/0/0/1
 area 1
  virtual-link 10.0.0.212
  interface TenGigE 0/2/0/00/0/0/0


router ospfv3 1
 router-id 10.0.0.212
 area 0
  interface TenGigE 0/3/0/10/0/0/1
 area 1
  virtual-link 10.0.0.217
  interface TenGigE 0/2/0/00/0/0/0
```

In this example, all interfaces on router ABR1 use MD5 authentication:

```
router ospf ABR1
 router-id 10.10.10.10
 authentication message-digest
```

```
 message-digest-key 100 md5 0 cisco
 area 0
  interface TenGigE 0/2/0/10/0/0/1
  interface TenGigE 0/3/0/00/0/0/0
 area 1
  interface TenGigE 0/2/0/00/0/0/0
  virtual-link 10.10.5.5
!
!
```

In this example, only area 1 interfaces on router ABR3 use MD5 authentication:

```
router ospf ABR2
 router-id 10.10.5.5
 area 0
 area 1
  authentication message-digest
  message-digest-key 100 md5 0 cisco
  interface TenGigE 0/9/0/10/0/0/1
  virtual-link 10.10.10.10
 area 3
  interface Loopback 0
  interface TenGigE 0/9/0/00/0/0/0
!
```

Summarize Subnetwork LSAs on 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 Do one of the following:

- **router ospf** *process-name*
- **router ospfv3** *process-name*

Example:

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

or

```
RP/0/RP0/CPU0:router(config)# router ospfv3 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

or

Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **router-id** { *router-id* }

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3
```

Configures a router ID for the OSPF process.

Note
We recommend using a stable IPv4 address as the router ID.

Step 4 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# area
```

Enters area configuration mode and configures a nonbackbone area for the OSPF process.

- The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 5 Do one of the following:

- **range** *ip-address mask* [**advertise** | **not-advertise**]
- **range** *ipv6-prefix / prefix-length* [**advertise** | **not-advertise**]

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# range 192.168.0.0 255.255.0.0 advertise
```

or

```
RP/0/RP0/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/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces to the area.

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

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

end —Prompts user to take one of these actions:

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

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 TenGigE 0/0/0/0
 area 1
  range 2300::/16
  interface TenGigE 0/0/0/0
```

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.

Redistribute Routes into OSPF

This task redistributes routes from an IGP (could be a different OSPF process) into OSPF.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 Do one of the following:

- **router ospf** *process-name*
- **router ospfv3** *process-name*

Example:

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

or

```
RP/0/RP0/CPU0:router(config)# router ospfv3 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

or

Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **router-id** { *router-id* }

Example:

```
RRP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3
```

Configures a router ID for the OSPF process.

Note
We recommend using a stable IPv4 address as the router ID.

Step 4 **redistribute** *protocol* [*process-id*] { **level-1** | **level-1-2** | **level-2** } [**metric** *metric-value*] [**metric-type** *type-value*] [**match** { **external** [**1** | **2**]} [**tag** *tag-value*] [**route-policy** *policy-name*]

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# redistribute bgp 100
```

or

```
RP/0/RP0/CPU0:router(config-router)#redistribute bgp 110
```

Redistributes OSPF routes from one routing domain to another routing domain.

or

Redistributes OSPFv3 routes from one routing domain to another routing domain.

- This command causes the router to become an ASBR by definition.
- OSPF tags all routes learned through redistribution as external.
- The protocol and its process ID, if it has one, indicate the protocol being redistributed into OSPF.
- The metric is the cost you assign to the external route. The default is 20 for all protocols except BGP, whose default metric is 1.
- The OSPF example redistributes BGP autonomous system 1, Level 1 routes into OSPF as Type 2 external routes.
- The OSPFv3 example redistributes BGP autonomous system 1, Level 1 and 2 routes into OSPF. The external link type associated with the default route advertised into the OSPFv3 routing domain is the Type 1 external route.

Note
RPL is not supported for OSPFv3.

Step 5 Do one of the following:

- **summary-prefix** *address mask* [**not-advertise**] [**tag** *tag*]
- **summary-prefix** *ipv6-prefix / prefix-length* [**not-advertise**] [**tag** *tag*]

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# summary-prefix 10.1.0.0 255.255.0.0
```

or

```
RP/0/RP0/CPU0:router(config-router)# summary-prefix 2010:11:22::/32
```

(Optional) Creates aggregate addresses for OSPF.

or

(Optional) Creates aggregate addresses for OSPFv3.

- This command provides external route summarization of the non-OSPF routes.
- External ranges that are being summarized should be contiguous. Summarization of overlapping ranges from two different routers could cause packets to be sent to the wrong destination.
- This command is optional. If you do not specify it, each route is included in the link-state database and advertised in LSAs.
- In the OSPFv2 example, the summary address 10.1.0.0 includes address 10.1.1.0, 10.1.2.0, 10.1.3.0, and so on. Only the address 10.1.0.0 is advertised in an external LSA.
- In the OSPFv3 example, the summary address 2010:11:22::/32 has addresses such as 2010:11:22:0:1000::1, 2010:11:22:0:2000:679:1, and so on. Only the address 2010:11:22::/32 is advertised in the external LSA.

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

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

end —Prompts user to take one of these actions:

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

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 TenGigE 0/0/0/0
```

Nonstop Forwarding for OSPF Version 2

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 the 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.

Configure 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 software image on the router. If an NSF-capable router discovers that it has non-NSF-aware neighbors on a particular network segment, it disables NSF capabilities for that segment. Other network segments composed entirely of NSF-capable or NSF-aware routers continue to provide NSF capabilities.

Note The following are restrictions when configuring nonstop forwarding:

- OSPF Cisco NSF for virtual links is not supported.
- Neighbors must be NSF aware.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router ospf** *process-name*

Example:

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

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **router-id** { *router-id* }

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3
```

Configures a router ID for the OSPF process.

Note

We recommend using a stable IPv4 address as the router ID.

Step 4 Do one of the following:

- **nsf cisco**
- **nsf cisco enforce global**

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# nsf cisco enforce global
```

Enables Cisco NSF operations for the OSPF process.

- Use the **nsf cisco** command without the optional **enforce** and **global** keywords to end the NSF restart mechanism on the interfaces of detected non-NSF neighbors and allow NSF neighbors to function properly.
- Use the **nsf cisco** command with the optional **enforce** and **global** keywords if the router is expected to perform NSF during restart. However, if non-NSF neighbors are detected, NSF restart is canceled for the entire OSPF process.

Step 5 **nsf interval** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# nsf interval 120
```

Sets the minimum time between NSF restart attempts.

Note
When you use this command, the OSPF process must be up for at least 90 seconds before OSPF attempts to perform an NSF restart.

Step 6 **nsfflush-delay-time***seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)#nsf flush-delay-time 1000
```

Sets the maximum time allowed for external route learning in seconds.

Step 7 **nsflifetime***seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)#nsf lifetime 90
```

Sets the maximum route lifetime of NSF following a restart in seconds.

Step 8 **nsfietf**

Example:

```
RP/0/RP0/CPU0:router(config-ospf)#nsf ietf
```

Enables ietf graceful restart.

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

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

end —Prompts user to take one of these actions:

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

OSPF Shortest Path First Throttling

OSPF SPF throttling makes it possible to configure SPF scheduling in millisecond intervals and to potentially delay SPF calculations during network instability. SPF is scheduled to calculate the Shortest Path Tree (SPT) when there is a change in topology. One SPF run may include multiple topology change events.

The interval at which the SPF calculations occur is chosen dynamically and based on the frequency of topology changes in the network. The chosen interval is within the boundary of the user-specified value ranges. If network topology is unstable, SPF throttling calculates SPF scheduling intervals to be longer until topology becomes stable.

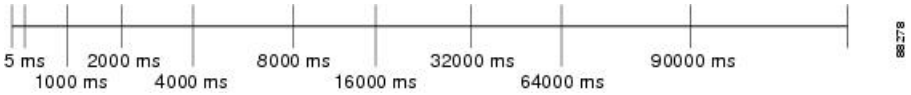
SPF calculations occur at the interval set by the **timers throttle spf** command. The wait interval indicates the amount of time to wait until the next SPF calculation occurs. Each wait interval after that calculation is twice as long as the previous interval until the interval reaches the maximum wait time specified.

The SPF timing can be better explained using an example. In this example, the start interval is set at 5 milliseconds (ms), initial wait interval at 1000 ms, and maximum wait time at 90,000 ms.

```
timers spf 5 1000 90000
```

Figure 3: SPF Calculation Intervals Set by the timers spf Command

This figure shows the intervals at which the SPF calculations occur as long as at least one topology change event is received in a given wait interval.

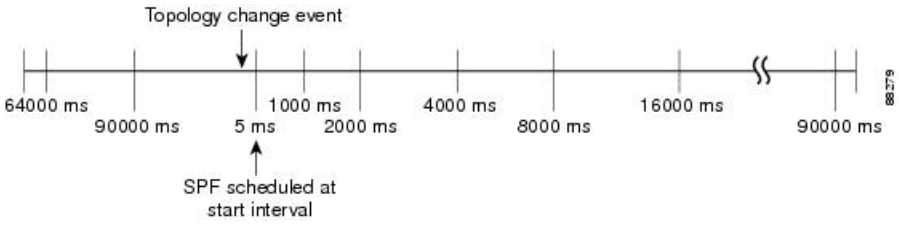


Notice that the wait interval between SPF calculations doubles when at least one topology change event is received during the previous wait interval. After the maximum wait time is reached, the wait interval remains the same until the topology stabilizes and no event is received in that interval.

If the first topology change event is received after the current wait interval, the SPF calculation is delayed by the amount of time specified as the start interval. The subsequent wait intervals continue to follow the dynamic pattern.

If the first topology change event occurs after the maximum wait interval begins, the SPF calculation is again scheduled at the start interval and subsequent wait intervals are reset according to the parameters specified in the **timers throttle spf** command. Notice in Figure 4: Timer Intervals Reset After Topology Change Event, on page 95 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 4: Timer Intervals Reset After Topology Change Event



Configure 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 Do one of the following:

- **router ospf** *process-name*
- **router ospfv3** *process-name*

Example:

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

or

```
RP/0/RP0/CPU0:router(config)# router ospfv3 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

or

Enables OSPFv3 routing for the specified routing process and places the router in router ospfv3 configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **router-id** { *router-id* }

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3
```

Configures a router ID for the OSPF process.

Note
We recommend using a stable IPv4 address as the router ID.

Step 4 **timers throttle spf** *spf-start spf-hold spf-max-wait*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# timers throttle spf 10 4800 90000
```

Sets SPF throttling timers.

Step 5 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)#  area 0
```

Enters area configuration mode and configures a backbone area.

- The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 6 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces to the area.

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

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

end —Prompts user to take one of these actions:

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

Step 8 Do one of the following:

- **show ospf** [*process-name*]
- **show ospfv3** [*process-name*]

Example:

```
RP/0/RP0/CPU0:router# show ospf 1
```

or

```
RP/0/RP0/CPU0:router# RP/0/RP0/CPU0:router# show ospfv3 2
```

(Optional) Displays SPF throttling timers.

Graceful Restart for OSPFv3

The OSPFv3 Graceful Shutdown feature preserves the data plane capability in these circumstances:

- 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.

Configure OSPFv3 Graceful Restart

This task explains how to configure a graceful restart for an OSPFv3 process. This task is optional.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router ospfv3** *process-name*

Example:

```
RP/0/RP0/CPU0:router(config)# router ospfv3 test
```

Enters router configuration mode for OSPFv3. The process name is a WORD that uniquely identifies an OSPF routing process. The process name is any alphanumeric string no longer than 40 characters without spaces.

Step 3 **graceful-restart**

Example:

```
RP/0/RP0/CPU0:router(config-ospfv3)#graceful-restart
```

Enables graceful restart on the current router.

Step 4 **graceful-restart lifetime**

Example:

```
RP/0/RP0/CPU0:router(config-ospfv3)# graceful-restart lifetime 120
```

Specifies a maximum duration for a graceful restart.

- The default lifetime is 95 seconds.

- The range is 90 to 3600 seconds.

Step 5 **graceful-restart interval** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ospfv3)# graceful-restart interval 120
```

Specifies the interval (minimal time) between graceful restarts on the current router.

- The default value for the interval is 90 seconds.
- The range is 90 to 3600 seconds.

Step 6 **graceful-restart helper disable**

Example:

```
RP/0/RP0/CPU0:router(config-ospfv3)# graceful-restart helper disable
```

Disables the helper capability.

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

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

end —Prompts user to take one of these actions:

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

Step 8 **show ospfv3** [*process-name* [*area-id*]] **database grace**

Example:

```
RP/0/RP0/CPU0:router# show ospfv3 1 database grace
```

Displays the state of the graceful restart link.

Display Information About Graceful Restart

This section describes the tasks you can use to display information about a graceful restart.

- To see if the feature is enabled and when the last graceful restart ran, use the **show ospf** command. To see details for an OSPFv3 instance, use the **show ospfv3** *process-name* [*area-id*] **database grace** command.

Displaying the State of the Graceful Restart Feature

The following screen output shows the state of the graceful restart capability on the local router:

```
RP/0/RP0/CPU0:router# show ospfv3 1 database grace
```

```
Routing Process "ospfv3 1" with ID 2.2.2.2
Initial SPF schedule delay 5000 msecs
Minimum hold time between two consecutive SPFs 10000 msecs
Maximum wait time between two consecutive SPFs 10000 msecs
Initial LSA throttle delay 0 msecs
Minimum hold time for LSA throttle 5000 msecs
Maximum wait time for LSA throttle 5000 msecs
Minimum LSA arrival 1000 msecs
LSA group pacing timer 240 secs
Interface flood pacing timer 33 msecs
Retransmission pacing timer 66 msecs
Maximum number of configured interfaces 255
Number of external LSA 0. Checksum Sum 00000000
Number of areas in this router is 1. 1 normal 0 stub 0 nssa
Graceful Restart enabled, last GR 11:12:26 ago (took 6 secs)
   Area BACKBONE(0)
       Number of interfaces in this area is 1
       SPF algorithm executed 1 times
       Number of LSA 6. Checksum Sum 0x0268a7
       Number of DCbitless LSA 0
       Number of indication LSA 0
       Number of DoNotAge LSA 0
       Flood list length 0
```

OSPFv2OSPF SPF Prefix Prioritization

The OSPFv2 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 OSPFv2OSPF 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 OSPFv2OSPF prefix installation into the global RIB during SPF.

SPF prefix prioritization is disabled by default. In disabled mode, /32 prefixes are installed into the global RIB, before other prefixes. If SPF prioritization is enabled, routes are matched against the route-policy criteria and are assigned to the appropriate priority queue based on the SPF priority set. Unmatched prefixes, including /32s, are placed in the low-priority queue.

If all /32s are desired in the high-priority queue or medium-priority queue, configure this single route map:

```
prefix-set ospf-medium-prefixes
  0.0.0.0/0 ge 32
  end-set
```

Configure OSPFv2 OSPF SPF Prefix Prioritization

Perform this task to configure OSPFv2 OSPF SPF (shortest path first) prefix prioritization.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
RP/0/RP0/CPU0:router(config)#prefix-set ospf-critical-prefixes
RP/0/RP0/CPU0:router(config-pfx)#66.0.0.0/16
RP/0/RP0/CPU0:router(config-pfx)#end-set
```

Configures the prefix set.

Step 3 **route-policy** *route-policy name* **if destination in** *prefix-set name* **then set spf-priority** {**critical** | **high** | **medium**} **endif**

Example:

```
RP/0/RP0/CPU0:router#route-policy ospf-spf-priority
RP/0/RP0/CPU0:router(config-rpl)#if destination in ospf-critical-prefixes then
 set spf-priority critical
endif
RP/0/RP0/CPU0:router(config-rpl)#end-policy
```

Configures route policy and sets OSPF SPF priority.

Step 4 Use one of these commands:

- **router ospf** *ospf-name*
- **router ospfv3** *ospfv3-name*

Example:

```
RP/0/RP0/CPU0:router# router ospf 1
```

Or

```
RP/0/RP0/CPU0:router# router ospfv3 1
```

Enters Router OSPF configuration mode.

Step 5 **router ospf** *ospf name*

Example:

```
RP/0/RP0/CPU0:router# router ospf 1
```

Enters Router OSPF configuration mode.

Step 6 **spf prefix-priority route-policy** *route-policy name*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# spf prefix-priority route-policy ospf-spf-priority
```

Or

```
RP/0/RP0/CPU0:router(config-ospfv3)#spf prefix-priority route-policy ospf3-spf-priority
```

Configures SPF prefix-priority for the defined route policy.

Note
Configure the **spf prefix-priority** command under router OSPF.

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

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

end —Prompts user to take one of these actions:

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

Step 8 **show rpl route-policy** *route-policy name* **detail**

Example:

```
RP/0/RP0/CPU0:router#show rpl route-policy ospf-spf-priority detail
  prefix-set ospf-critical-prefixes
    66.0.0.0/16
  end-set
  !
  route-policy ospf-spf-priority
    if destination in ospf-critical-prefixes then
      set spf-priority critical
    endif
  end-policy
  !
```

Displays the set SPF prefix priority.

OSPFv2

OSPFv3

This example shows how to configure /32 prefixes as medium-priority, in general, in addition to placing some /32 and /24 prefixes in critical-priority and high-priority queues:

```
prefix-set ospf-critical-prefixes
  192.41.5.41/32,
  11.1.3.0/24,
  192.168.0.44/32
  end-set
  !
  prefix-set ospf-high-prefixes
  44.4.10.0/24,
  192.41.4.41/32,
  41.4.41.41/32
  end-set
  !
  prefix-set ospf-medium-prefixes
  0.0.0.0/0 ge 32
  end-set
  !
```

```
  route-policy ospf-priority
    if destination in ospf-high-prefixes then
      set spf-priority high
    else
      if destination in ospf-critical-prefixes then
        set spf-priority critical
      else
        if destination in ospf-medium-prefixes then
          set spf-priority medium
        endif
      endif
    endif
  end-policy


router ospf 1
   spf prefix-priority route-policy ospf-priority
   area 0
    interface TenGigE 0/0/0/1
    !
   !
   area 3
    interface TenGigE 0/0/0/0
    !
   !
   area 8
    interface TenGigE 0/0/0/0


router ospfv3 1
   spf prefix-priority route-policy ospf-priority
   area 0
    interface TenGigE 0/0/0/1
    !
   !
   area 3
    interface TenGigE 0/0/0/0
    !
   !
   area 8
    interface TenGigE 0/0/0/0
```

Configure OSPF as a Provider Edge to Customer Edge (PE-CE) Protocol

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router ospf** *process-name*

Example:

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

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# vrf vrf1
```

Creates a VRF instance and enters VRF configuration mode.

Step 4 **router-id** { *router-id* }

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# router-id 192.168.4.3
```

Configures a router ID for the OSPF process.

Note
We recommend using a stable IPv4 address as the router ID.

Step 5 **redistribute** *protocol* [*process-id*] { | |} [**metric** *metric-value*] [**metric-type** *type-value*] [**match** { **external** [**1** | **2**] }] [**tag** *tag-value*] **route-policy** *policy-name*]

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# redistribute bgp 1 metric 2
```

Redistributes OSPF 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 example shows the redistribution of BGP autonomous system 1, Level 1 routes into OSPF as Type 2 external routes.

Step 6 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# area 0
```

Enters area configuration mode and configures an area for the OSPF process.

- The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area.

Step 7 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces to the VRF.

Step 8 exit

Example:

```
RP/0/RP0/CPU0:router(config-if)# exit
```

Exits interface configuration mode.

Step 9 **domain-id** *[* **secondary** *]* **type** *{* **0005** | **0105** | **0205** | **8005** *}* **value** *value*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# domain-id type 0105 value 1AF234
```

Specifies the OSPF VRF domain ID.

- The *value* argument is a six-octet hex number.

Step 10 **domain-tag** *tag*

Example:

```
RP/0/RP0/CPU0:router(config-0spf-vrf)# domain-tag 234
```

Specifies the OSPF VRF domain tag.

- The valid range for *tag* is 0 to 4294967295.

Step 11 disable-dn-bit-check

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# disable-dn-bit-check
```

Specifies that down bits should be ignored.

Step 12 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Create 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router ospf** *process-name*

Example:

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

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)#  area 0
```

Enters area configuration mode and configures a backbone area.

- The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 4 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces to the area.

Step 5 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# exit
```

Enters OSPF configuration mode.

Step 6 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# vrf vrf1
```

Creates a VRF instance and enters VRF configuration mode.

Step 7 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# area 0
```

Enters area configuration mode and configures an area for a VRF instance under the OSPF process.

- 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 8 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces to the VRF.

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

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

end —Prompts user to take one of these actions:

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

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.

Configure 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router ospf** *process-name*

Example:

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

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **mpls ldp auto-config**

Example:

```
RP/0/RP0/CPU0:router(config-ospf)#  mpls ldp auto-config
```

Enables LDP IGP interface auto-configuration for an OSPF instance.

- Optionally, this command can be configured for an area of an OSPF instance.

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

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

end —Prompts user to take one of these actions:

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

Configure 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router ospf** *process-name*

Example:

```
RP/0/RP0/CPU0:router(config)# router ospf 100
```

Identifies the OSPF routing process and enters OSPF configuration mode.

Step 3 (Optional) **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# vrf red
```

Specifies the non-default VRF.

Step 4 Use one of the following commands:

- **mpls ldp sync**
- **area** *area-id* **mpls ldp sync**
- **area** *area-id* **interface** *name* **mpls ldp sync**

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# mpls ldp sync
```

Enables LDP IGP synchronization on an interface.

Step 5 (Optional) Use one of the following commands:

- **mpls ldp sync**
- **area** *area-id* **mpls ldp sync**
- **area** *area-id* **interface** *name* **mpls ldp sync**

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# mpls ldp sync
```

```
RP/0/RP0/CPU0:router(config-ospf-vrf)# area 1 mpls ldp sync
```

Enables LDP IGP synchronization on an interface for the specified VRF.

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

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

end —Prompts user to take one of these actions:

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

Step 7 (Optional) **show mpls ldp vrf** *vrf-name* **ipv4 igp sync**

Example:

```
RP/0/RP0/CPU0:router# show mpls ldp vrf red ipv4 igp sync
```

Displays the LDP IGP synchronization information for the specified VRF for address family IPv4.

Step 8 (Optional) **show mpls ldp vrf all ipv4 igp sync**

Example:

```
RP/0/RP0/CPU0:router# show mpls ldp vrf all ipv4 igp sync
```

Displays the LDP IGP synchronization information for all VRFs for address family IPv4.

Step 9 (Optional) **show mpls ldp** { **ipv4** | **ipv6** }**igp sync**

Example:

```
RP/0/RP0/CPU0:router# show mpls ldp ipv4 igp sync
```

```
RP/0/RP0/CPU0:router# show mpls ldp ipv6 igp sync
```

Displays the LDP IGP synchronization information for IPv4 or IPv6 address families.

Example

The example shows how to configure LDP IGP synchronization for OSPF.

```
router ospf 100
mpls ldp sync
!
mpls ldp
```

```
 igp sync delay 30
!
```

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.

Configure 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router ospf** *process-name*

Example:

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

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **router-id** { *router-id* }

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# router id 192.168.4.3
```

Configures a router ID for the OSPF process.

Note
We recommend using a stable IPv4 address as the router ID.

Step 4 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# area 1
```

Enters area configuration mode.

The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces to the area.

Step 6 **authentication** [**message-digest** *keychain* | **keychain** *keychain* | **null**]

Configures an MD5 keychain. Keychains can be configured at the following three levels of authentication:

- Router-level authentication
- Area-level authentication
- Interface-level authentication

Example:

The following example shows the configuration for keychain authentication.

```
RP/0/RP0/CPU0:router(config-ospf)# authentication keychain test_chain  -----□ Router-level
 authentication
RP/0/RP0/CPU0:router(config-ospf)# router-id 1.1.1.1
RP/0/RP0/CPU0:router(config-ospf)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-ospf)# area 1
RP/0/RP0/CPU0:router(config-ospf-ar)# authentication keychain test_chain  -----□ Area-level
 authentication
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/1
RP/0/RP0/CPU0:router(config-ospf-ar-if)# authentication keychain test_chain ---□
Interface-level authentication
RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit
```

Example:

The following example shows the configuration for message-digest authentication.

```
RP/0/RP0/CPU0:router(config-ospf-ar-if)# authentication message-digest keychain ospf_int1
```

Note
In the above example, the *ospf_intl* keychain must be configured before you attempt this step.

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

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

end —Prompts user to take one of these actions:

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

Examples

The following example shows how to configure the keychain *ospf_intf_1* that contains five key IDs. Each key ID is configured with different **send-lifetime** values; however, all key IDs specify the same text string for the key.

```
key chain ospf_intf_1
key 1
send-lifetime 11:30:30 May 1 2007 duration 600
cryptographic-algorithm MD5
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 TenGigE 0/0/0/0 interface:

```
show ospf 1 interface TenGigE 0/0/0/0

TenGigE 0/0/0/0 is up, line protocol is up
  Internet Address 100.10.10.2/24, Area 0
  Process ID 1, Router ID 2.2.2.1, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 2.2.2.1, Interface address 100.10.10.2
  Backup Designated router (ID) 1.1.1.1, Interface address 100.10.10.1
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:02
  Index 3/3, flood queue length 0
  Next 0(0)/0(0)
  Last flood scan length is 2, maximum is 16
  Last flood scan time is 0 msec, maximum is 0 msec
```

```
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 1.1.1.1  (Backup Designated Router)
  Suppress hello for 0 neighbor(s)
  Keychain-based authentication enabled
    Key id used is 3
  Multi-area interface Count is 0
```

The following example shows output for configured keys that are active:

```
show key chain ospf_intf_1

 Key-chain: ospf_intf_1/ -

 Key 1 -- text "0700325C4836100B0314345D"
   cryptographic-algorithm -- MD5
   Send lifetime:   11:30:30, 01 May 2007 - (Duration) 600
   Accept lifetime: Not configured
 Key 2 -- text "10411A0903281B051802157A"
   cryptographic-algorithm -- MD5
   Send lifetime:   11:40:30, 01 May 2007 - (Duration) 600
   Accept lifetime: Not configured
 Key 3 -- text "06091C314A71001711112D5A"
   cryptographic-algorithm -- MD5
   Send lifetime:   11:50:30, 01 May 2007 - (Duration) 600  [Valid now]
   Accept lifetime: Not configured
 Key 4 -- text "151D181C0215222A3C350A73"
   cryptographic-algorithm -- MD5
   Send lifetime:   12:00:30, 01 May 2007 - (Duration) 600
   Accept lifetime: Not configured
 Key 5 -- text "151D181C0215222A3C350A73"
   cryptographic-algorithm -- MD5
   Send lifetime:   12:10:30, 01 May 2007 - (Duration) 600
   Accept lifetime: Not configured
```

GTSM TTL Security Mechanism for OSPF

OSPF is a link state protocol that requires networking devices to detect topological changes in the network, flood Link State Advertisement (LSA) updates to neighbors, and quickly converge on a new view of the topology. However, during the act of receiving LSAs from neighbors, network attacks can occur, because there are no checks that unicast packets are originating from a neighbor that is one hop away or multiple hops away over virtual links.

For virtual links, OSPF packets travel multiple hops across the network; hence, the TTL value can be decremented several times. For these type of links, a minimum TTL value must be allowed and accepted for multiple-hop packets.

To filter network attacks originating from invalid sources traveling over multiple hops, the Generalized TTL Security Mechanism (GTSM), RFC 3682, is used to prevent the attacks. GTSM filters link-local addresses and allows for only one-hop neighbor adjacencies through the configuration of TTL value 255. The TTL value in the IP header is set to 255 when OSPF packets are originated, and checked on the received OSPF packets against the default GTSM TTL value 255 or the user configured GTSM TTL value, blocking unauthorized OSPF packets originated from TTL hops away.

Configure Generalized TTL Security Mechanism (GTSM) for OSPF

This task explains how to set the security time-to-live mechanism on an interface for GTSM.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router ospf** *process-name*

Example:

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

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Note
The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3 **router-id** { *router-id* }

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# router id 10.10.10.100
```

Configures a router ID for the OSPF process.

Note
We recommend using a stable IPv4 address as the router ID.

Step 4 **log adjacency changes** [**detail** | **disable**]

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar-if)# log adjacency changes detail
```

(Optional) Requests notification of neighbor changes.

- By default, this feature is enabled.
- The messages generated by neighbor changes are considered notifications, which are categorized as severity Level 5 in the **logging console** command. The **logging console** command controls which severity level of messages are sent to the console. By default, all severity level messages are sent.

Step 5 **nsf** { **cisco** [**enforce global**] | **ietf** [**helper disable**]}

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# nsf ietf
```

(Optional) Configures NSF OSPF protocol.

The example enables graceful restart.

Step 6 **timers throttle spf** *spf-start spf-hold spf-max-wait*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# timers throttle spf 500 500 10000
```

(Optional) Sets SPF throttling timers.

Step 7 **area** *area-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf)# area 1
```

Enters area configuration mode.

The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 8 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE0/0/0/0
```

Enters interface configuration mode and associates one or more interfaces to the area.

Step 9 **security ttl** [**disable** | **hops** *hop-count*]

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar-if)# security ttl hops 2
```

Sets the security TTL value in the IP header for OSPF packets.

Step 10 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

Step 11 **show ospf** [*process-name*] [*area-id*] **interface** [*type interface-path-id*]

Example:

```
RP/0/RP0/CPU0:router# show ospf 1 interface TenGigE0/0/0/0
```

Displays OSPF interface information.

Example

The following is sample output that displays the GTSM security TTL value configured on an OSPF interface:

```
show ospf 1 interface  TenGigE0/5/0/0

 TenGigE0/0/0/0 is up, line protocol is up
   Internet Address 120.10.10.1/24, Area 0
   Process ID 1, Router ID 100.100.100.100, Network Type BROADCAST, Cost: 1
   Transmit Delay is 1 sec, State BDR, Priority 1
   TTL security enabled, hop count 2
   Designated Router (ID) 102.102.102.102, Interface address 120.10.10.3
   Backup Designated router (ID) 100.100.100.100, Interface address 120.10.10.1
   Flush timer for old DR LSA due in 00:02:36
   Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
     Hello due in 00:00:05
   Index 1/1, flood queue length 0
   Next 0(0)/0(0)
   Last flood scan length is 1, maximum is 4
   Last flood scan time is 0 msec, maximum is 0 msec
   Neighbor Count is 1, Adjacent neighbor count is 1
     Adjacent with neighbor 102.102.102.102  (Designated Router)
   Suppress hello for 0 neighbor(s)
   Multi-area interface Count is 0
```

OSPF Penalty for Link Delay Anomaly

Note For information on configuring the link delay anomaly threshold values, refer to Link Anomaly Detection with IGP Penalty in the Segment Routing Configuration Guide.

When you configure Link Anomaly Detection in SR-PM, PM sets an anomaly bit (A-bit). When IGP receives the A-bit, IGP can automatically increase the IGP metric of the link by a user-defined amount (IGP penalty). This updated IGP metric is advertised in the network to make this link undesirable or unusable. When the link recovers, PM resets the A-bit.

Note When node is reloaded, the default or configured IGP metric (without penalty) is advertised until a new measurement is available.

Configuration

```
RP/0/RSP0/CPU0:ios(config)# router ospf 1
RP/0/RSP0/CPU0:ios(config-ospf)# area 0
RP/0/RSP0/CPU0:ios(config-ospf-ar)# interface GigabitEthernet 0/2/0/0
RP/0/RSP0/CPU0:ios(config-ospf-ar-if)# cost-fallback anomaly delay igp-metric increment 25
```

Running Configuration

```
router ospf 1
 area 0
  interface GigabitEthernet0/2/0/0
   cost-fallback anomaly delay igp-metric increment 25
  !
 !
```

Limiting LSA Numbers in a OSPF Link-State Database

Table 10: Feature History Table

Feature Name	Release	Description
Limiting LSA numbers in a OSPF Link-State Database	Release 7.9.1	The nonself-generated link-state advertisements (LSAs) for a given Open Shortest Path First (OSPF) process is limited to 500000. This protection mechanism prevents routers from receiving many LSAs, preventing CPU failure and memory shortages, and is enabled by default from this release onwards. If you have over 500000 LSAs in your network, configure the **max-lsa** command with the expected LSA scale before upgrading to this release or later. This feature modifies the following commands: • **show ospf** to display the maximum number of redistributed prefixes. • **show ospf database database-summary detail** to display the number of LSA counts per router. • **show ospf database database-summary adv-router** *router ID* to display the router information and the LSAs received from a particular router.

The OSPF Link-State Database Overload Protection feature allows you to protect the OSPF routing process by limiting the number of nonself-generated link-state advertisements (LSAs) for a given Open Shortest Path First (OSPF) process. When other routers in the network have been misconfigured, they may generate a high volume of LSAs. This mechanism prevents routers from receiving a large number of LSAs, thereby preventing

CPU failure and memory shortages. With this feature, the router keeps a count of the number of nonself-generated LSAs it has received.

Note The max-lsa limit was not enabled by default before Release 7.9.1. Starting from Release 7.9.1, this command is enabled by default and the default limit of the nonself-generated LSA is set at 500000. If you have more than 500000 LSAs in a network, you must configure the **max-lsa** command with the expected LSA scale before upgrading to Release 7.9.1 or above.

Restriction

This feature is supported only on OSPFv2 and not on OSPFv3.

System output messages

The range of nonself-generated LSA allowed is 1-4294967294. The threshold percentage to log warning is 75%. The system log message is generated every 5% above the default or configured threshold value until 100% is reached.

When the number of LSAs reaches or exceeds the threshold limit, the router displays the following logs:

When number of LSAs exceed threshold value

```
%ROUTING-OSPF-4-MAX_LSA_THR : Reached threshold (60% [configured: 60%])
for maximum number of non self-generated LSAs in vrf "default" - LSA (max:
1000 cur: 600)
```

When number of LSAs exceed maximum limit

```
%ROUTING-OSPF-1-MAX_LSA : Maximum number of non self-generated LSAs
exceeded in vrf "default" - LSA (max: 1000, cur: 1001)
```

When OSPF instance ignores all adjacencies for ignore-time period if the number of LSAs exceed the limit

```
%ROUTING-OSPF-2-MAX_LSA_IGNORE_ENTER : Max LSA exceeded in vrf "default".
Adjacencies will be kept down for 5 minutes
```

When OSPF instance tries to recover the adjacencies after ignore-time period

```
%ROUTING-OSPF-6-MAX_LSA_IGNORE_EXIT : Max-lsa ignore timed out in vrf
"default". Adjacencies will be brought up by accepting and sending hellos
```

When the ignore count is exceeded on the OSPF instance

```
%ROUTING-OSPF-1-MAX_LSA_PERM_IGNORE : Max-lsa ignore count exceeded in
vrf "default" - Staying in ignore state. Restart or Clear OSPF process
to recover
```

When number of LSAs exceed threshold or limit the top contributing routers information will be displayed

```
%ROUTING-OSPF-2-MAX_LSA_RTR_INFO : Top 1 LSA contributor in vrf "default".
RTR:192.168.0.4   Total:498   Type3:0   Type5:492   Type7:0   Type10:0
Type11:6   Others:0
```

The following commands displays the LSA counts:

- **show ospf database database-summary detail** command displays the number of LSA counts per router sorted by total LSA count.

```
Router#show ospf database database-summary detail


            OSPF Router with ID (192.168.0.1) (Process ID 1)

Router 192.168.0.4 LSA summary
  LSA Type      Count    Delete   Maxage
  Router        0        0        0
  Network       0        0        0
  Summary Net   0        0        0
  Summary ASBR  0        0        0
  Type-5 Ext    697      0        0
  Type-7 Ext    0        0        0
  Opaque Link   0        0        0
  Opaque Area   0        0        0
  Opaque AS     6        0        0
  Total         703      0        0

Router 192.168.0.1 LSA summary
  LSA Type      Count    Delete   Maxage
  Router        1        0        0
  Network       0        0        0
  Summary Net   0        0        0
  Summary ASBR  0        0        0
  Type-5 Ext    0        0        0
  Type-7 Ext    0        0        0
  Opaque Link   0        0        0
  Opaque Area   64       0        0
  Opaque AS     0        0        0
  Total         65       0        0

Router 192.168.0.2 LSA summary
  LSA Type      Count    Delete   Maxage
  Router        1        0        0
  Network       0        0        0
  Summary Net   21       0        0
  Summary ASBR  2        0        0
  Type-5 Ext    0        0        0
  Type-7 Ext    0        0        0
  Opaque Link   0        0        0
  Opaque Area   21       0        0
  Opaque AS     0        0        0
  Total         45       0        0

Router 192.168.0.6 LSA summary
  LSA Type      Count    Delete   Maxage
  Router        1        0        0
  Network       0        0        0
  Summary Net   21       0        0
  Summary ASBR  2        0        0
  Type-5 Ext    0        0        0
  Type-7 Ext    0        0        0
  Opaque Link   0        0        0
  Opaque Area   19       0        0
  Opaque AS     0        0        0
  Total         43       0        0

Router 192.168.0.3 LSA summary
  LSA Type      Count    Delete   Maxage
  Router        0        0        0
```

```
  Network       0         0         0
  Summary Net   0         0         0
  Summary ASBR  0         0         0
  Type-5 Ext    7         0         0
  Type-7 Ext    0         0         0
  Opaque Link   0         0         0
  Opaque Area   0         0         0
  Opaque AS     6         0         0
  Total         13        0         0
```

- **show ospf database database-summary adv-router *router ID*** command displays the router information and the LSAs received from the particular router.

```
Router#show ospf database database-summary adv-router 192.168.0.4


            OSPF Router with ID (192.168.0.1) (Process ID 1)

Router 192.168.0.4 LSA summary
  LSA Type      Count    Delete   Maxage
  Router        0        0        0
  Network       0        0        0
  Summary Net   0        0        0
  Summary ASBR  0        0        0
  Type-5 Ext    697      0        0
  Type-7 Ext    0        0        0
  Opaque Link   0        0        0
  Opaque Area   0        0        0
  Opaque AS     6        0        0
  Total         703      0        0
```

Limiting the Maximum Redistributed Type-3 LSA Prefixes in OSPF

Table 11: Feature History Table

Feature Name	Release	Description
Limiting the Maximum Redistributed Type-3 LSA Prefixes in OSPF	Release 7.9.1	By default, the maximum redistributed Type-3 LSA prefixes for a given OSPF process is now limited to 100000. This mechanism prevents OSPF from redistributing a large number of prefixes as Type-3 LSAs and therefore preventing high CPU utilization and memory shortages. Once the number of redistributed prefixes is reached or exceeds the threshold value, the system log message is generated, and no more prefixes are redistributed. Support was added to count only the number of redistributed prefixes instead of the number of LSAs. Individual prefixes are counted when multiple prefixes are aggregated in an LSA using the **summary-prefix** command.

Redistribution allows different routing protocols to exchange routing information. This is used to allow connectivity to span multiple routing protocols. Open Shortest Path First (OSPF) supports a user-defined maximum number of prefixes (routes) that are allowed to be redistributed into OSPF from other protocols or other OSPF instances.

Prior to Release 7.9.1, the maximum redistributed-prefixes limit was applied only to those prefixes that are redistributed as Type-5 and Type-7 LSAs. Starting from Release 7.9.1, the maximum redistributed-prefixes limit is also applied to the prefixes that are redistributed as Type-3 LSAs. The maximum redistributed Type-3 LSA prefixes for a given OSPF process is limited to 100000.

If the router redistributes more than 10000 prefixes as Type 3, 5, or 7 LSAs, then you must configure a higher limit using the **maximum redistributed-prefixes** command.

Starting from Release 7.9.1, if the **redistribute protocol lsa-type summary** command is configured to redistribute the routes from particular protocol as Type-3 LSAs, then those Type-3 LSAs are accounted for maximum redistributed prefixes.

System output messages

The range of prefixes that are redistributed as Type-3 LSAs is 1-4294967295. The threshold percentage to log warning is 75%. The system log message is generated every 5% above the default or configured threshold value until 100% is reached.

When the number of LSAs reaches or exceeds the threshold limit, the router displays the following logs:

The redistributed prefixes count reached the maximum limit

```
%ROUTING-OSPF-4-REDIST_THR_PFX : Reached Redistribution prefix threshold
in vrf "default", current (70%) 700 prefixes, limit  1000
```

The redistributed prefixes count exceeds the threshold percentage

```
%ROUTING-OSPF-1-REDIST_MAX_PFX : Redistribution prefix limit has been
reached in vrf "default" - current 1000 prefixes, limit 1000
```

The redistributed prefixes count falls below the threshold percentage

```
%ROUTING-OSPF-5-REDIST_MAX_PFX_RECOVER : Recovered from Redistribution
limit-hit scenario in vrf "default", prefix count less than threshold -
current (69%) 699 prefixes, limit 1000
```

The **show ospf** command displays the maximum number of redistributed prefixes, which is configured at 1000.

```
Router #show ospf
Thu Dec  8 18:16:48.332 IST

 Routing Process "ospf 1" with ID 192.168.0.1
 Role: Primary Active
 NSR (Non-stop routing) is Enabled
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 It is an autonomous system boundary router
 Maximum number of non self-generated LSA allowed 1000
    Current number of non self-generated LSA 804
    Threshold for warning message 60%
    Ignore-time 1 minutes, reset-time 2 minutes
    Ignore-count allowed 2, current ignore-count 0
 Redistributing External Routes from,
    static
 Maximum number of redistributed prefixes 1000
    Threshold for warning message 70%
    Current number of redistributed prefixes 100
```

Autonomous System Boundary Router Isolation and Adjacency Control for LSA Overflows

Table 12: Feature History Table

Feature Name	Release Name	Description
Autonomous System Boundary Router Isolation and Adjacency Control for LSA Overflows	Release 7.10.1	In a network employing an Autonomous System Boundary Router (ASBR) and other routers, you are now assured of uninterrupted traffic flow even if the ASBR generates LSAs that exceed the limit you configured. This is made possible as you can now isolate ASBRs and also control the duration of adjacency in the EXCHANGE or LOADING phase. By isolating the ASBR from its immediate neighbors, the remaining network topology can continue to function without disruption, effectively preventing any adverse impact on traffic flow. This approach also simplifies the recovery process, as manual intervention is only necessary for the immediate neighbors of the ASBR routers. This feature introduces these changes: **CLI:** • **max-external-lsa** • **exchange-timer** **YANG Data Model:** • `Cisco-IOS-XR-ipv4-ospf-cfg.yang` • `Cisco-IOS-XR-ipv4-ospf-oper.yang` • `Cisco-IOS-XR-um-router-ospf-cfg.yang` (see GitHub, YANG Data Models Navigator)

Overview

The Autonomous System Boundary Routers (ASBR) Isolation and Adjacency Control for LSA Overflows feature prevents LSAs generated by a misbehaving ASBR from flooding the autonomous system. This, in turn, averts disruptions in the OSPF network. The feature accomplishes this by restricting external LSAs from immediate neighbors and by limiting the duration of adjacency in the EXCHANGE or LOADING phase.

The default BGP routing table is distributed in OSPF when there is a need to propagate the default route to OSPF routers. This occurs when OSPF routers are configured to advertise default routes to their OSPF neighbors or when the OSPF routers receive a default route from their BGP peers. The need for using ACL or RPL in OSPF arises when specific filtering, modification, or manipulation is required.

The misconfiguration of ASBR can occur in several scenarios. These include instances where a filter mechanism, such as an Access Control List (ACL) or Routing Policy Language (RPL) policy, is missing. Additionally, misconfiguration can arise when the configured filter mechanism fails due to reasons like human errors during rule creation or modification. Unintended overlaps or similar issues can also lead to ASBR misconfiguration.

The misconfiguration in the ASBR leads to the unintended redistribution of the default BGP routing table in OSPF. As a result, the ASBR generates millions of LSAs, of either Type 5 or Type 7, within the OSPF domain causing an unexpected flood of external LSAs. These LSAs flood throughout the entire autonomous system triggering route calculations on all routers causing the depletion of the available resources, such as high memory and CPU usage. This causes a strain on the system. The sustained strain ultimately results in the failure of the router. As a result, the entire OSPF domain may experience a shutdown, causing disruptions.

This feature facilitates the isolation of a misbehaving ASBR if the number of LSAs exceed a predetermined threshold. The immediate neighbors of the misbehaving ASBR isolate it from themselves. Hence, the rest of the network continues uninterrupted operation. The feature mitigates the potential loss of network traffic and ensure stability of the network. If another ASBR exists, which correctly leaks only relevant routes into the OSPF domain, it ensures uninterrupted connectivity. This feature simplifies the recovery process as manual interruption is limited to the immediate neighbors of the ASBR routers.

The neighboring routers constantly monitor the influx of type 5 and type 7 LSAs originating from the peer ASBR. Once the predetermined threshold is exceeded, the neighboring router proceeds to terminate the adjacency with the misbehaving ASBR. This action effectively halts the communication link between the neighboring router and the misbehaving ASBR, ensuring compliance with the imposed limit.

The neighbor maintains the adjacency in a deactivated state for a specified interval. After the interval elapses, the neighbor initiates the process of forming the adjacency again.

When the ASBR rectifies the issue with the origination of the external LSA, the neighbor establishes the adjacency successfully. Alternatively, if the neighbor disrupts the adjacency, user intervention is required to enable ASBR to rectify the redistribution issue. If the ASBR successfully rectifies the redistribution issue, the adjacency recovers automatically. If the ASBR fails to rectify the redistribution issue, it results in the repetition of the cycle. However, it’s not recommended for the system to attempt retries indefinitely. Therefore, this feature incorporates a counter to monitor the number of times the adjacency towards the ASBR is unsuccessful due to reaching the configured maximum limit for external LSAs. Once the count reaches a specified number of attempts, the neighbor ceases further efforts to establish the adjacency, resulting in a permanent adjacency downtime.

After reaching the permanent adjacency downtime, manual intervention is necessary to restore the adjacency. This involves shutting down the interface and after bringing it back up by issuing the **no shutdown** command.

Restrictions for ASBR Isolation and Adjacency Control for LSA Overflows

- All neighbors of the ASBR must support this feature for the protection mechanism to function effectively. If any neighbor fails to support this feature, that neighbor leaks external LSAs to the rest of the OSPF domain.
- To add a second level of protection, enable the **max-lsa** command on the routers within the OSPF domain. For more information on enabling a second level of protection, see "Limiting LSA numbers in a OSPF Link-State Database".

Configure ASBR Isolation and Adjacency Control for LSA Overflows

Configuration Example

- Configure the timers value for the maximum limit for external LSAs.
- Configure the threshold percentage for maximum limit for external LSA using the **exchange-timer** command.
- Configure the redistribution of BGP into OSPF domain using the **redistribute bgp** command.

```
Router(config)# router ospf 1
Router(config-ospf)# log adjacency changes detail
Router(config-ospf)# router-id 203.0.113.1
Router(config-ospf)# maximum redistributed-prefixes 150 90
Router(config-ospf)# exchange-timer 3 hold-time 2 recovery-count 5
Router(config-ospf)# max-external-lsa 100 75
Router(config-ospf)# redistribute bgp 100
```

Running Configuration

```
router ospf 1
  log adjacency changes detail
  router-id 203.0.113.1
  maximum redistributed-prefixes 150 90
  exchange-timer 3 hold-time 2 recovery-count 5
  max-external-lsa 100 75
  redistribute bgp 100
```

Verification

Verify the configuration of the maximum number of external LSAs and the exchange timer. The output displays the list of ASBRs that went down because the number of LSAs exceeded the configured maximum number.

```
Router# show ospf

 Routing Process "ospf 1" with ID 203.0.113.1
 Role: Primary Active
 NSR (Non-stop routing) is Enabled
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 It is an area border and autonomous system boundary router
 Redistributing External Routes from,
```

```
    bgp 100
 Maximum number of redistributed prefixes 150
    Threshold for warning message 90%
    Current number of redistributed prefixes 0
Maximum of external prefixes per router 100 (suppress-neighbor)
 Warning threshold 75%
 2 ASBR(s) have hit the limit now and links made down:
      192.168.0.5
      192.168.0.6
Maximum Exchange time 3 Hold time 2 Max Recovery count 5
Time remaining for suppressed-asbr stale-checker to expire: 00:03:48
```

Verify the state of the OSPF interface. The show output displays the status of adjacency hold timer and exchange timer. The output indicates that the exchange timer is recovering from being down due to exceeding the configured limit of maximum external LSAs.

```
Router# show ospf interface

TenGigE0/0/0/3.1 is up, line protocol is up
  Internet Address 203.0.113.1/24, Area 0, SID 0, Strict-SPF SID 0

Adjacency hold timer expired last : Mar 28 10:44:28.510
  Exchange timer expired last : Mar 28 10:47:38.077
  Exchange timer status: Recovering after max external LSA
    Exchange timer is running, time remaining: 00:08:29
    Recovery count: 1
```

The show command output shows the ASBRs that went down because they exceeded the configured limit of maximum external LSAs.

```
Router# show ospf suppressed-asbrs

Suppressed ASBR(s) for ospf 1 with ID 203.0.113.1

ASBR : 209.165.200.254
  ASBR down since : Apr  9 18:32:07.257
  Interface(s):
    Area 0                TenGigE0/0/0/3.3
      Adjacency hold timer expired last : Never
      Exchange timer expired last : Never
      Exchange timer status: Down due to max external LSA
      Hold timer is running, time remaining: 00:01:52
      Recovery count 0
    Area 1                TenGigE0/0/0/3.1
      Adjacency hold timer expired last : Never
      Exchange timer expired last : Never
      Exchange timer status: Down due to max external LSA
      Hold timer is running, time remaining: 00:01:52
      Recovery count 0
```

Optimization of OSPF to Reduce TCAM Usage by LPTS

Starting Cisco IOS XR Software Release 24.2.1, OSPF optimization has been implemented to enhance the efficiency of OSPF traffic management on routers. This feature aims to conserve TCAM resources by consolidating OSPF flow entries across multiple interfaces within the same address family, while excluding specific interface types such as GRE tunnels due to their distinct requirements. The optimization takes into

account configurations like TTL settings, which can influence OSPF traffic handling. Hence, this upgrade makes it possible for routers to handle more OSPF connections and offer more features without having to add new hardware.

Restrictions for Optimization of OSPF to reduce TCAM Usage by LPTS:

The following are the restrictions for optimization of OSPF to reduce TCAM usage by LPTS:

- Post optimization, OSPF sessions has only one set KNOWN flow entries for all interfaces per address family except GRE TUNNEL interfaces.
- If the TTL configuration is applied on the OSPF interface's inclusion in the session, the interface is excluded from optimization.
- Interface ID is not be visible in the **show lpts pifib hardware entry breif location** *LC node ID* command for any interface except GRE tunnel interfaces.
- Each GRE tunnel interface has its own set of KNOWN entries and is not part of optimisation.

For more information on LPTS, see *Implementing LPTS* chapter in *IP Addresses and Services Configuration Guide.*

IGP link state

IGP Link-State Database Distribution

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

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

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

Enhance Network Level Isolation

-

Autonomous System Boundary Router Isolation and Adjacency Control for LSA Overflows

Table 13: Feature History Table

Feature Name	Release Name	Description
Autonomous System Boundary Router Isolation and Adjacency Control for LSA Overflows	Release 7.10.1	In a network employing an Autonomous System Boundary Router (ASBR) and other routers, you are now assured of uninterrupted traffic flow even if the ASBR generates LSAs that exceed the limit you configured. This is made possible as you can now isolate ASBRs and also control the duration of adjacency in the EXCHANGE or LOADING phase. By isolating the ASBR from its immediate neighbors, the remaining network topology can continue to function without disruption, effectively preventing any adverse impact on traffic flow. This approach also simplifies the recovery process, as manual intervention is only necessary for the immediate neighbors of the ASBR routers. This feature introduces these changes: **CLI:** • **max-external-lsa** • **exchange-timer** **YANG Data Model:** • `Cisco-IOS-XR-ipv4-ospf-cfg.yang` • `Cisco-IOS-XR-ipv4-ospf-oper.yang` • `Cisco-IOS-XR-um-router-ospf-cfg.yang` (see GitHub, YANG Data Models Navigator)

Overview

The Autonomous System Boundary Routers (ASBR) Isolation and Adjacency Control for LSA Overflows feature prevents LSAs generated by a misbehaving ASBR from flooding the autonomous system. This, in turn, averts disruptions in the OSPF network. The feature accomplishes this by restricting external LSAs from immediate neighbors and by limiting the duration of adjacency in the EXCHANGE or LOADING phase.

The default BGP routing table is distributed in OSPF when there is a need to propagate the default route to OSPF routers. This occurs when OSPF routers are configured to advertise default routes to their OSPF neighbors or when the OSPF routers receive a default route from their BGP peers. The need for using ACL or RPL in OSPF arises when specific filtering, modification, or manipulation is required.

The misconfiguration of ASBR can occur in several scenarios. These include instances where a filter mechanism, such as an Access Control List (ACL) or Routing Policy Language (RPL) policy, is missing. Additionally, misconfiguration can arise when the configured filter mechanism fails due to reasons like human errors during rule creation or modification. Unintended overlaps or similar issues can also lead to ASBR misconfiguration.

The misconfiguration in the ASBR leads to the unintended redistribution of the default BGP routing table in OSPF. As a result, the ASBR generates millions of LSAs, of either Type 5 or Type 7, within the OSPF domain causing an unexpected flood of external LSAs. These LSAs flood throughout the entire autonomous system triggering route calculations on all routers causing the depletion of the available resources, such as high memory and CPU usage. This causes a strain on the system. The sustained strain ultimately results in the failure of the router. As a result, the entire OSPF domain may experience a shutdown, causing disruptions.

This feature facilitates the isolation of a misbehaving ASBR if the number of LSAs exceed a predetermined threshold. The immediate neighbors of the misbehaving ASBR isolate it from themselves. Hence, the rest of the network continues uninterrupted operation. The feature mitigates the potential loss of network traffic and ensure stability of the network. If another ASBR exists, which correctly leaks only relevant routes into the OSPF domain, it ensures uninterrupted connectivity. This feature simplifies the recovery process as manual interruption is limited to the immediate neighbors of the ASBR routers.

The neighboring routers constantly monitor the influx of type 5 and type 7 LSAs originating from the peer ASBR. Once the predetermined threshold is exceeded, the neighboring router proceeds to terminate the adjacency with the misbehaving ASBR. This action effectively halts the communication link between the neighboring router and the misbehaving ASBR, ensuring compliance with the imposed limit.

The neighbor maintains the adjacency in a deactivated state for a specified interval. After the interval elapses, the neighbor initiates the process of forming the adjacency again.

When the ASBR rectifies the issue with the origination of the external LSA, the neighbor establishes the adjacency successfully. Alternatively, if the neighbor disrupts the adjacency, user intervention is required to enable ASBR to rectify the redistribution issue. If the ASBR successfully rectifies the redistribution issue, the adjacency recovers automatically. If the ASBR fails to rectify the redistribution issue, it results in the repetition of the cycle. However, it’s not recommended for the system to attempt retries indefinitely. Therefore, this feature incorporates a counter to monitor the number of times the adjacency towards the ASBR is unsuccessful due to reaching the configured maximum limit for external LSAs. Once the count reaches a specified number of attempts, the neighbor ceases further efforts to establish the adjacency, resulting in a permanent adjacency downtime.

After reaching the permanent adjacency downtime, manual intervention is necessary to restore the adjacency. This involves shutting down the interface and after bringing it back up by issuing the **no shutdown** command.

References for 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.

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.

After the router boots up, it sends OSPFv2 traps only after two dead intervals have passed to prevent flooding caused by frequent state changes.

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

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 the software, the parameters of interfaces in the same area can be exclusively configured with a single command, or parameter values can be inherited from a higher hierarchical level—such as from the area configuration level or the router ospf configuration levels.

For example, the hello interval value for an interface is determined by this precedence “IF” statement:

If the **hello interval** command is configured at the interface configuration level, then use the interface configured value, else

If the **hello interval** command is configured at the area configuration level, then use the area configured value, else

If the **hello interval** command is configured at the router ospf configuration level, then use the router ospf configured value, else

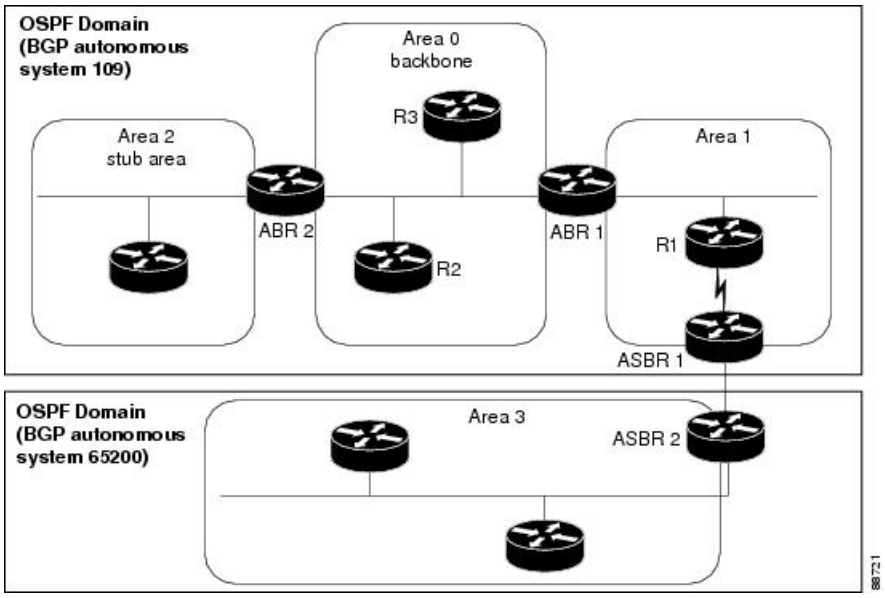
Use the default value of the command.

OSPF Routing Components

Before implementing OSPF, you must know what the routing components are and what purpose they serve. They consist of the autonomous system, area types, interior routers, ABRs, and ASBRs.

Figure 5: OSPF Routing Components

This figure illustrates the routing components in an OSPF network topology.



Autonomous Systems

The autonomous system is a collection of networks, under the same administrative control, that share routing information with each other. An autonomous system is also referred to as a routing domain. *Figure 1: OSPF Routing Components* 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 the *Figure 1: OSPF Routing Components*, autonomous system 109 consists of three areas: Area 0, Area 1, and Area 2.

OSPF hides the topology of an area from the rest of the autonomous system. The network topology for an area is visible only to routers inside that area. When OSPF routing is within an area, it is called *intra-area routing*. This routing limits the amount of link-state information flood into the network, reducing routing traffic. It also reduces the size of the topology information in each router, conserving processing and memory requirements in each router.

Also, the routers within an area cannot see the detailed network topology outside the area. Because of this restricted view of topological information, you can control traffic flow between areas and reduce routing traffic when the entire autonomous system is a single routing domain.

Backbone Area

A backbone area is responsible for distributing routing information between multiple areas of an autonomous system. OSPF routing occurring outside of an area is called *interarea routing*.

The backbone itself has all properties of an area. It consists of ABRs, routers, and networks only on the backbone. As shown in *Figure 1: OSPF Routing Components*, Area 0 is an OSPF backbone area. Any OSPF backbone area has a reserved area ID of 0.0.0.0.

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 1: OSPF Routing Components* section, 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 1: OSPF Routing Components*) 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 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
- Broadcast networks

You can configure your 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 the *OSPFAuthentication Message Digest Management* section.

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.

OSPF FIB Download Notification

OSPF FIB Download Notification feature minimizes the ingress traffic drop for a prolonged period of time after the line card reloads.

Open Shortest Path First (OSPF) registers with Routing Information Base (RIB) through ITAL which keeps the interface down until all the routes are downloaded to Forwarding Information Base (FIB). OSPF gets the Interface Up notification when all the routes on the reloaded line card are downloaded through RIB/FIB.

RIB provides notification to registered clients when a:

- Node is lost.
- Node is created.
- Node's FIB upload is completed.

Designated Router (DR) for OSPF

On broadcast or NBMA segments only, OSPF minimizes the amount of information being exchanged on a segment by choosing one router to be a DR and one router to be a BDR. Thus, the routers on the segment have a central point of contact for information exchange. Instead of each router exchanging routing updates with every other router on the segment, each router exchanges information with the DR and BDR. The DR and BDR relay the information to the other routers.

The software looks at the priority of the routers on the segment to determine which routers are the DR and BDR. The router with the highest priority is elected the DR. If there is a tie, then the router with the higher router ID takes precedence. After the DR is elected, the BDR is elected the same way. A router with a router priority set to zero is ineligible to become the DR or BDR.

Default Route for OSPF

Type 5 (ASE) LSAs are generated and flooded to all areas except stub areas. For the routers in a stub area to be able to route packets to destinations outside the stub area, a default route is injected by the ABR attached to the stub area.

The cost of the default route is 1 (default) or is determined by the value specified in the **default-cost** command.

Link-State Advertisement Types for OSPF Version 2

Each of the following LSA types has a different purpose:

- Router LSA (Type 1)—Describes the links that the router has within a single area, and the cost of each link. These LSAs are flooded within an area only. The LSA indicates if the router can compute paths based on quality of service (QoS), whether it is an ABR or ASBR, and if it is one end of a virtual link. Type 1 LSAs are also used to advertise stub networks.
- Network LSA (Type 2)—Describes the link state and cost information for all routers attached to a multiaccess network segment. This LSA lists all the routers that have interfaces attached to the network segment. It is the job of the designated router of a network segment to generate and track the contents of this LSA.
- Summary LSA for ABRs (Type 3)—Advertises internal networks to routers in other areas (interarea routes). Type 3 LSAs may represent a single network or a set of networks aggregated into one prefix. Only ABRs generate summary LSAs.
- Summary LSA for ASBRs (Type 4)—Advertises an ASBR and the cost to reach it. Routers that are trying to reach an external network use these advertisements to determine the best path to the next hop. ABRs generate Type 4 LSAs.
- Autonomous system external LSA (Type 5)—Redistributes routes from another autonomous system, usually from a different routing protocol into OSPF.
- Autonomous system external LSA (Type 7)—Provides for carrying external route information within an NSSA. Type 7 LSAs may be originated by and advertised throughout an NSSA. NSSAs do not receive or originate Type 5 LSAs. Type 7 LSAs are advertised only within a single NSSA. They are not flooded into the backbone area or into any other area by border routers.
- Intra-area-prefix LSAs (Type 9)—A router can originate multiple intra-area-prefix LSAs for every router or transit network, each with a unique link-state ID. The link-state ID for each intra-area-prefix LSA describes its association to either the router LSA or network LSA and contains prefixes for stub and transit networks.
- Area local scope (Type 10)—Opaque LSAs are not flooded past the borders of their associated area.
- Link-state (Type 11)—The LSA is flooded throughout the AS. The flooding scope of Type 11 LSAs are equivalent to the flooding scope of AS-external (Type 5) LSAs. Similar to Type 5 LSAs, the LSA is rejected if a Type 11 opaque LSA is received in a stub area from a neighboring router within the stub area. Type 11 opaque LSAs have these attributes:
  - LSAs are flooded throughout all transit areas.
  - LSAs are not flooded into stub areas from the backbone.
  - LSAs are not originated by routers into their connected stub areas.

Link-State Advertisement Types for OSPFv3

Each of the following LSA types has a different purpose:

- Router LSA (Type 1)—Describes the link state and costs of a the router link to the area. These LSAs are flooded within an area only. The LSA indicates whether the router is an ABR or ASBR and if it is one end of a virtual link. Type 1 LSAs are also used to advertise stub networks. In OSPFv3, these LSAs have no address information and are network protocol independent. In OSPFv3, router interface information may be spread across multiple router LSAs. Receivers must concatenate all router LSAs originated by a given router before running the SPF calculation.
- Network LSA (Type 2)—Describes the link state and cost information for all routers attached to a multiaccess network segment. This LSA lists all OSPF routers that have interfaces attached to the network segment. Only the elected designated router for the network segment can generate and track the network LSA for the segment. In OSPFv3, network LSAs have no address information and are network-protocol-independent.
- Interarea-prefix LSA for ABRs (Type 3)—Advertises internal networks to routers in other areas (interarea routes). Type 3 LSAs may represent a single network or set of networks aggregated into one prefix. Only ABRs generate Type 3 LSAs. In OSPFv3, addresses for these LSAs are expressed as “prefix and prefix length” instead of “address and mask.” The default route is expressed as a prefix with length 0.
- Interarea-router LSA for ASBRs (Type 4)—Advertises an ASBR and the cost to reach it. Routers that are trying to reach an external network use these advertisements to determine the best path to the next hop. ABRs generate Type 4 LSAs.
- Autonomous system external LSA (Type 5)—Redistributes routes from another autonomous system, usually from a different routing protocol into OSPF. In OSPFv3, addresses for these LSAs are expressed as “prefix and prefix length” instead of “address and mask.” The default route is expressed as a prefix with length 0.
- Autonomous system external LSA (Type 7)—Provides for carrying external route information within an NSSA. Type 7 LSAs may be originated by and advertised throughout an NSSA. NSSAs do not receive or originate Type 5 LSAs. Type 7 LSAs are advertised only within a single NSSA. They are not flooded into the backbone area or into any other area by border routers.
- Link LSA (Type 8)—Has link-local flooding scope and is never flooded beyond the link with which it is associated. Link LSAs provide the link-local address of the router to all other routers attached to the link or network segment, inform other routers attached to the link of a list of IPv6 prefixes to associate with the link, and allow the router to assert a collection of Options bits to associate with the network LSA that is originated for the link.
- Intra-area-prefix LSAs (Type 9)—A router can originate multiple intra-area-prefix LSAs for every router or transit network, each with a unique link-state ID. The link-state ID for each intra-area-prefix LSA describes its association to either the router LSA or network LSA and contains prefixes for stub and transit networks.

An address prefix occurs in almost all newly defined LSAs. The prefix is represented by three fields: Prefix Length, Prefix Options, and Address Prefix. In OSPFv3, addresses for these LSAs are expressed as “prefix and prefix length” instead of “address and mask.” The default route is expressed as a prefix with length 0.

Inter-area-prefix and intra-area-prefix LSAs carry all IPv6 prefix information that, in IPv4, is included in router LSAs and network LSAs. The Options field in certain LSAs (router LSAs, network LSAs, interarea-router LSAs, and link LSAs) has been expanded to 24 bits to provide support for OSPF in IPv6.

In OSPFv3, the sole function of link-state ID in interarea-prefix LSAs, interarea-router LSAs, and autonomous system external LSAs is to identify individual pieces of the link-state database. All addresses or router IDs that are expressed by the link-state ID in OSPF Version 2 are carried in the body of the LSA in OSPFv3.

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.

Modes of Graceful Restart Operation

The operational modes that a router can be in for this feature are restart mode and helper moderestart mode, helper mode, and protocol shutdown 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.

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.

If OSPFv3 attempts a restart too soon after the most recent restart, the OSPFv3 process is most likely crashing repeatedly, so the new graceful restart stops running. To control the period between allowable graceful restarts, use the **graceful-restart interval** command. When OSFPv3 starts a graceful restart with the first interface that comes up, a timer starts running to limit the duration (or lifetime) of the graceful restart. You can configure this period with the **graceful-restart lifetime** command. On each interface that comes up, a *grace* LSA (Type 11) is flooded to indicate to the neighboring routers that this router is attempting graceful restart. The neighbors enter into helper mode. The designated router and backup designated router check of the hello packet received from the restarting neighbor is bypassed, because it might not be valid.

Helper Mode

Helper mode is enabled by default. When a helper router receives a grace LSA (Type 11) from a router that is attempting a graceful restart, the following events occur:

- If helper mode has been disabled through the **graceful-restart helper disable** command, the router drops the LSA packet.
- If helper mode is enabled, the router enters helper mode if all of the following conditions are met:
  - The local router itself is not attempting a graceful restart.
  - The local (helping) router has full adjacency with the sending neighbor.
  - The value of *lsage* (link state age) in the received LSA is less than the requested grace period.
  - The sender of the grace LSA is the same as the originator of the grace LSA.

- Upon entering helper mode, a router performs its helper function for a specific period of time. This time period is the lifetime value from the router that is in restart mode—minus the value of *lsage* in the received grace LSA. If the graceful restart succeeds in time, the helper's timer is stopped before it expires. If the helper's timer does expire, the adjacency to the restarting router is brought down, and normal OSPFv3 functionality resumes.
- The dead timer is not honored by the router that is in helper mode.
- A router in helper mode ceases to perform the helper function in any of the following cases:
  - The helper router is able to bring up a FULL adjacency with the restarting router.
  - The local timer for the helper function expires.

Protocol Shutdown Mode

In this mode the OSPFv3 operation is completely disabled. This is accomplished by flushing self-originated link state advertisements (LSAs), immediately bringing down local OSPFv3-supported interfaces, and clearing the Link State Database (LSDB). The non-local LSDB entries are removed by OSPFv3, These are not flooded (MaxAged).

The protocol shutdown mode can be invoked either manually through the **protocol shutdown** command that disables the protocol instance or when the OSPFv3 process runs out of memory. These events occur when protocol shut down is performed:

- The local Router LSA and all local Link LSAs are flushed. All other LSAs are eventually aged out by other OSPFv3 routers in the domain.
- OSPFv3 neighbors not yet in Full state with the local router are brought down with the Kill_Nbr event.
- After a three second delay, empty Hello packets are immediately sent to each neighbor that has an active adjacency.
  - An empty Hello packet is sent periodically until the dead_interval has elapsed.
  - When the dead_interval elapses, Hello packets are no longer sent.

After a Dead Hello interval delay (4 X Hello Interval), the following events are then performed:

- The LSA database from that OSPFv3 instance is cleared.
- All routes from RIB that were installed by OSPFv3 are purged.

The router will not respond to any OSPF control packets it receives from neighbors while in protocol shutdown state.

Protocol Restoration

The method of restoring the protocol is dependent on the trigger that originally invoked the shut down. If the OSPFv3 was shut down using the **protocol shutdown** command, then use the **no protocol shutdown** command to restore OSPFv3 back to normal operation. If the OSPFv3 was shutdown due to a Critical Memory message from the sysmon, then a Normal Memory message from sysmon, which indicates that sufficient memory has been restored to the processor, restores the OSPFv3 protocol to resume normal operation. When OSPFv3 is shutdown due to the Critical Memory trigger, it must be manually restarted when normal memory levels are restored on the route processor. It will not automatically restore itself.

These events occur when the OSPFv3 is restored:

1. All OSPFv3 interfaces are brought back up using the Hello packets and database exchange.
2. The local router and link LSAs are rebuilt and advertised.
3. The router replies normally to all OSPFv3 control messages received from neighbors.
4. Routes learned from other OSPFv3 routers are installed in RIB.

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.

Path Computation Element for OSPFv2

A PCE is an entity (component, application, or network node) that is capable of computing a network path or route based on a network graph and applying computational constraints.

PCE is accomplished when a PCE address and client is configured for MPLS-TE. PCE communicates its PCE address and capabilities to OSPF then OSPF packages this information in the PCE Discovery type-length-value (TLV) (Type 2) and reoriginates the RI LSA. OSPF also includes the Router Capabilities TLV (Type 1) in all its RI LSAs. The PCE Discovery TLV contains the PCE address sub-TLV (Type 1) and the Path Scope Sub-TLV (Type 2).

The PCE Address Sub-TLV specifies the IP address that must be used to reach the PCE. It should be a loop-back address that is always reachable, this TLV is mandatory, and must be present within the PCE Discovery TLV. The Path Scope Sub-TLV indicates the PCE path computation scopes, which refers to the PCE ability to compute or participate in the computation of intra-area, inter-area, inter-AS or inter-layer TE LSPs.

PCE extensions to OSPFv2 include support for the Router Information Link State Advertisement (RI LSA). OSPFv2 is extended to receive all area scopes (LSA Types 9, 10, and 11). However, OSPFv2 originates only area scope Type 10.

For detailed information for the Path Computation Element feature see the *Implementing MPLS Traffic Engineering* module of the *MPLS Configuration guide* and the following IETF drafts:

- draft-ietf-ospf-cap-09
- draft-ietf-pce-disco-proto-ospf-00

Management Information Base (MIB) for OSPFv3

Cisco IOS XR supports full MIBs and traps for OSPFv3, as defined in RFC 5643. The RFC 5643 defines objects of the Management Information Base (MIB) for use with the Open Shortest Path First (OSPF) Routing Protocol for IPv6 (OSPF version 3).

The OSPFv3 MIB implementation is based on the IETF draft *Management Information Base for OSPFv3 (draft-ietf-ospf-ospfv3-mib-8)*. Users need to update the NMS application to pick up the new MIB when upgraded to RFC 5643.

Multiple OSPFv3 Instances

SNMPv3 supports "contexts" that can be used to implement MIB views on multiple OSPFv3 instances, in the same system.

VRF-lite Support for OSPFv2

VRF-lite capability is enabled for OSPF version 2 (OSPFv2). VRF-lite is the virtual routing and forwarding (VRF) deployment without the BGP/MPLS based backbone. In VRF-lite, individual provider edge (PE) routers are directly connected using VRF interfaces. To enable VRF-lite in OSPFv2, configure the **capability vrf-lite** command in VRF configuration mode. When VRF-lite is configured, the DN bit processing and the automatic Area Border Router (ABR) status setting are disabled.

OSPFv3 Timers Update

The Open Shortest Path First version 3 (OSPFv3) timers link-state advertisements (LSAs) and shortest path first (SPF) throttle default values are updated to:

- **timers throttle lsa all**—*start-interval*: 50 milliseconds and *hold-interval*: 200 milliseconds
- **timers throttle spf** —*spf-start*: 50 milliseconds, *spf-hold*: 200 milliseconds, *spf-max-wait*: 5000 milliseconds

CHAPTER 3

IGP Flexible Algorithm in IP Networks

Flexible Algorithm allows operators to customize IGP shortest path computation according to their own needs. The Flexible Algorithm allows Link State IGPs (OSPF, ISIS) to compute paths using various constraints. Each Flexible Algorithm represents the triplet (Calculation-Type, Metric-Type, Constraints) which all routers in the area consistently elect using the defined selection algorithm.

In IP networks, the router uses IGP Flexible Algorithm by computing the paths to the IPv4 address [RFC0791] and IPv6 address [RFCS8200].

Each interface may be associated with one or more IP addresses, and each IP address may be associated with one Flexible Algorithm.

- Packets sent to an address that is associated with a Flexible Algorithm follow the constraint-based path as calculated by the Flexible Algorithm.
- Packet sent to an address, that is *not* associated with a Flexible Algorithm, follow the IGP least-cost path to the egress node.

IGP IP Flexible Algorithm allows forwarding of the voice traffic and data traffic over different paths in the IP network. It can, for example, provide a low latency path for the voice traffic.

This document describes the IS-IS protocol extensions to support IGP Flexible Algorithm in IP networks.

- Prerequisites for IP Flexible Algorithm, on page 144
- Flexible Algorithm Definition, on page 144
- Flexible Algorithm Definition Advertisement, on page 144
- IP Flexible Algorithm Prefix Advertisement, on page 144
- IP Flexible Algorithm Participation, on page 144
- Computing IP Flexible Algorithm Paths, on page 145
- IP Flexible Algorithm Forwarding, on page 145
- Flexible Algorithm Configuration, on page 145
- Associating the IP Address to Flexible Algorithm, on page 147
- Example: Configuring IS-IS IP Flexible Algorithm, on page 147
- Verifying IP Flexible Algorithm, on page 148
- Protecting Flexible Algorithm IP Prefixes, on page 151
- Example: Enabling Flexible Algorithm Protection, on page 151
- Flexible-Algorithm Redistribution in IP Networks, on page 156

Prerequisites for IP Flexible Algorithm

Configure the **data-plane ip** command under IGP flex-algo sub-mode to enable participation of the router with the IP Flexible Algorithm.

Flexible Algorithm Definition

The set consisting of (a) calculation-type, (b) metric-type and (c) a set of constraints is referred to as a Flexible-Algorithm Definition.

Flexible-Algorithm is a numeric identifier in the range 128-255 that is associated via provisioning with the Flexible-Algorithm Definition.

To guarantee the loop free forwarding for paths computed for a particular Flex-Algorithm, all routers that are configured to participate in a particular Flex-Algorithm, and in the same Flex-Algorithm definition advertisement scope must agree on the definition of the Flex-Algorithm as in ietf-lsr-flex-algo.

Flexible Algorithm Definition Advertisement

To guarantee the loop free forwarding for paths computed for a particular Flexible Algorithm, all routers in the network must share the same definition of the Flexible Algorithm. This is achieved by dedicated router(s) advertising the definition of each Flexible Algorithm. Such advertisement is associated with the priority to make sure that all routers agree on a single and consistent definition for each Flexible Algorithm.

Definition of Flexible Algorithm includes:

- Metric type
- Affinity constraints
- Exclude SRLG constraints

To enable the router to advertise the definition for the particular Flexible Algorithm, **advertise-definition** command is used. At least one router in the area, preferably two for redundancy, must advertise the Flexible Algorithm definition. Without the valid definition being advertised, the Flexible Algorithm are not be functional.

IP Flexible Algorithm Prefix Advertisement

The IPv4 and IPv6 Algorithm Prefix Reachability TLVs defined in draft-ietf-lsr-ip-flexalgo are used to advertise prefix reachability associated with a Flexible Algorithm.

IP Flexible Algorithm Participation

Each application using the Flexible Algorithm must use its own participation signaling. IP Flexible Algorithm uses ISIS IP Algorithm Sub-TLV as specified in draft-ietf-lsr-ip-flexalgo.

Computing IP Flexible Algorithm Paths

Each Flexible Algorithm maintains a separate set of paths—One set per each data-plane.

The IP Flexible Algorithm computation uses only the IP Flex-Algorithm prefixes that are advertised in IPv4 and IPv6 Algorithm Prefix Reachability TLV.

Note The performance impact is proportional to the number of IP Flexible Algorithms in the participating router. Routers using the algorithm may use additional CPU cycles to:

- Process new TLVs.
- Calculate the primary and backup paths for the IP Flex-Algorithm prefixes.
- Advertise the IPv4 or IPv6 Algorithm Prefix Reachability TLVs.

IP Flexible Algorithm Forwarding

IP Flexible Algorithm uses the base IPv4 and IPv6 packets for forwarding.

IP Flexible Algorithm prefixes are advertised in IGPs. The forwarding plane installs the IP Flex-Algorithm prefixes advertised by the receiving routers participating in the associated topology and algorithm.

The IP Flex-Algorithm prefixes can be protected by local LFA. When the prefix is associated with an algorithm, the LFA paths to such a prefix are calculated using the Flexible Algorithm in the associated topology. Thus, ensuring that the algorithm follows the same constraints as the calculation of the primary paths.

The IP Flex-Algorithm prefixes can be protected by TI-LFA. For more information on protecting the IP Flex-Algorithm prefixes, see Protecting Flexible Algorithm IP Prefixes, on page 151.

Flexible Algorithm Configuration

The following workflow enables IP Flexible Algorithm:

1. Configure the IP data-plane to participate in the Flexible Algorithm. See Enabling the Flexible Algorithm Participation, on page 146.
2. Define the parameters for FAD. See Flexible Algorithm Definitions, on page 146.
3. Advertise the definition. See Flexible Algorithm Advertisement, on page 146.
4. Define the admin groups. See Configuring Affinity, on page 146.
5. Verify the Flexible Algorithm configuration.
6. Associate the IP address of the interface to the Flexible Algorithm. See Associating the IP Address to Flexible Algorithm, on page 147.

Enabling the Flexible Algorithm Participation

The following command in IS-IS flex-algo submode configuration enables Flexible Algorithm participation on the native IP data plane:

```
router isis instance
flex-algo algo
data-plane ip
```

Note Segment Routing is the default data-plane. To use the IP data-plane, you must enable the **data-plane ip** command.

Flexible Algorithm Definitions

The following commands configure the Flexible Algorithm definition under the flex-algo submode configuration:

- **router isis** *instance*
  flex-algo *algo*
  metric-type {**delay** | **te**}
- **affinity** {**include-any** | **include-all** | **exclude-any**} *name1, name2, …*

  name—Name of the affinity map.
- **priority** *priority value*

  priority value—Priority used during the Flexible Algorithm definition election.

Flexible Algorithm Advertisement

The following command enables advertisement of the Flexible Algorithm definition in IS-IS:

```
router isis instance
flex-algo algo
advertise-definition
```

Configuring Affinity

The following command defines the affinity-map. Affinity-map associates the name with the particular Bit positions in the Extended Admin Group bitmask.

```
router isis instance
affinity-map name bit-position bit number
```

- *Name*—Name of the affinity-map.
- *bit number*—Bit position in the Extended Admin Group bitmask.

The following command associates the Flexible Algorithm-specific affinities with an interface:

```
router isis instance
interface type interface-path-id
affinity flex-algo name 1, name 2, …
```

name—Name of the affinity-map.

Associating the IP Address to Flexible Algorithm

To associate the Flexible Algorithm with the interface global IPv4 or IPv6 address, use the following commands:

- **ipv4 address** *prefix-length* [**secondary** | **algorithm** *algo-no*]
- **ipv6 address** *prefix-length* **algorithm** *algo-no*]

Note If you do not associate an algorithm, the default algorithm value (0) is associated.

The following example shows how to associate IPv4 address with a Flexible Algorithm for loopback interface 1:

```
Router(config)# interface loopback 1
Router(config-if#) ipv4 address 1.1.1.1/32
Router(config-if) #ipv4 address 1.1.1.1/32 algorithm 128 ----------> adds the algo value
128
Router(config-if)#commit
```

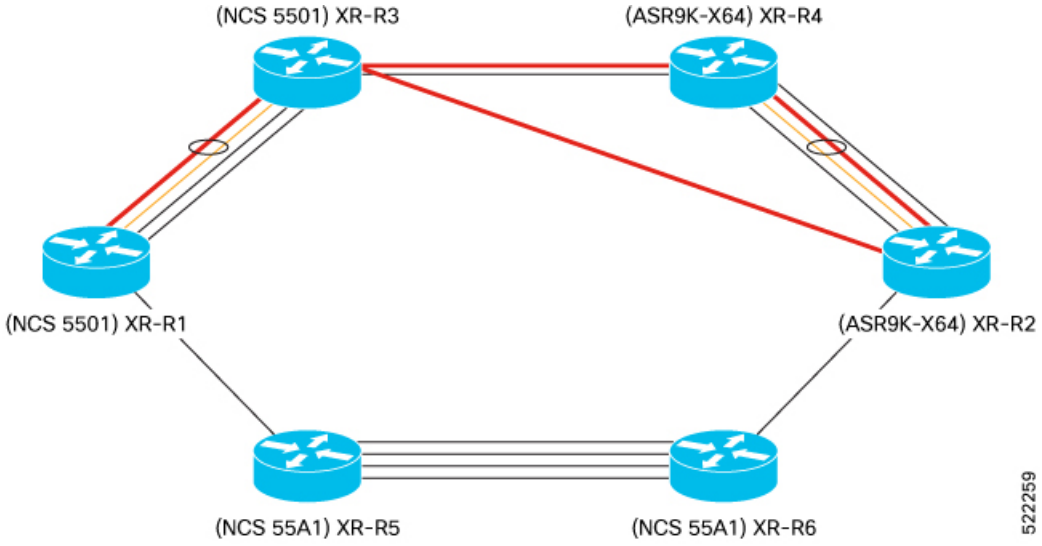
The following example associates the IPv6 address with a Flexible Algorithm for loopback interface 0:

```
Router(config)# interface loopback 0
Router(config-if)# ipv6 address 1::1/64 ---------------> add ipv6 add without algo
Router(config-if)# ipv6 address 1::1/64 algorithm 128 --------> add algo value 128
Router(config-if)#commit
```

Example: Configuring IS-IS IP Flexible Algorithm

The following figure is a six-node topology from R1 to R6 with configuration.

- Node R2 defines the FAD.
- IP Flexible Algorithm on node R2 ► R4 ► R3 ►R1.



This example shows the Flexible Algorithm 151 associated to IPv4 address at Loopback interface 51.

```
RP/0/RSP0/CPU0:Router# show run router isis
router isis 1
 is-type level-2-only
 net 49.0000.0000.0000.0012.00
 affinity-map RED bit-position 4
flex-algo 151
  data-plane ip
  metric-type delay
  advertise-definition
  affinity include-any RED
 !
!
 interface Loopback51
  passive
  address-family ipv4 unicast
  !
  address-family ipv6 unicast
```

This example shows the IP address association with the Flexible Algorithm on Loopback interface 51.

```
RP/0/RSP0/CPU0:Router# show running interface Loopback 51
Mon Nov 15 14:02:09.832 IST
interface Loopback51
 ipv4 address 2.2.2.51 255.255.255.255 algorithm 151
 ipv6 address 2:2:2::51/128 algorithm 151
!
RP/0/RSP0/CPU0:Router# show running interface HundredGigE 0/1/0/0.301
Mon Nov 15 14:02:38.800 IST
interface HundredGigE0/1/0/0.301
 ipv4 address 200.3.2.1 255.255.255.0 secondary algorithm 151
 ipv6 address 200::3:2:1/112 algorithm 151
 encapsulation dot1q 301
```

Verifying IP Flexible Algorithm

Use the **show isis flex-algo** *algo-no* to verify data-plane used with Flexible Algorithm configuration.

```
RP/0/RSP0/CPU0:RouterR2# show isis flex-algo 151
Mon Nov 15 15:54:06.291 IST

IS-IS 1 Flex-Algo Database

Flex-Algo 151:

  Level-2:
    Definition Priority: 128
    Definition Source: Router.00, (Local)
    Definition Equal to Local: Yes
    Definition Metric Type: Delay
    Definition Flex-Algo Prefix Metric: No
    Exclude Any Affinity Bit Positions:
    Include Any Affinity Bit Positions:  4
    Include All Affinity Bit Positions:
    Exclude SRLGs:
    Disabled: No

  Local Priority: 128
  FRR Disabled: No
  Microloop Avoidance Disabled: No

  Data Plane Segemnt Routing: No
  Data Plane IP: Yes
```

This example verifies the IP Algo Prefix Advertisements in ISIS database on R2.

```
RP/0/RP0/CPU0:RouterR1# show isis database detail verbose RouterR2
Mon Nov 15 16:05:22.188 IST

IS-IS 1 (Level-2) Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime/Rcvd  ATT/P/OL
RouterR2.00-00          0x0000004c   0xd97e        773  /1200         0/0/0
  Area Address:   49.0000
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 2.2.2.31/32 D:0 Metric: 0 Algorithm: 131
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
    Source Router ID: 2.2.2.2
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 2.2.2.51/32 D:0 Metric: 0 Algorithm: 151
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
    Source Router ID: 2.2.2.2
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 200.3.1.0/24 D:0 Metric: 10 Algorithm: 131

    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 200.3.2.0/24 D:0 Metric: 10 Algorithm: 151

    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv6 Algo Prefix: MT (IPv6 Unicast) 2:2:2::31/128 D:0 Metric: 0 Algorithm: 131
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv6 Algo Prefix: MT (IPv6 Unicast) 2:2:2::51/128 D:0 Metric: 0 Algorithm: 151
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  NLPID:          0xcc
  NLPID:          0x8e
  IP Address:     2.2.2.2
  Router ID:      2.2.2.2
  Metric: 0          IP-Extended 2.2.2.2/32
    Prefix-SID Index: 2, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix-SID Index: 102, Algorithm:131, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix Attribute Flags: X:0 R:0 N:1 E:0 A:0
    Source Router ID: 2.2.2.2
  Metric: 0          MT (IPv6 Unicast) IPv6 2:2:2::2/128
    Prefix-SID Index: 12, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix-SID Index: 112, Algorithm:131, R:0 N:1 P:0 E:0 V:0 L:0
```

This example verifies Flexible Algorithm 151 is present in only the topology—IP data-plane

```
RP/0/RP0/CPU0:RouterR1# show isis topology flex-algo 151 data-plane segment-routing
Mon Nov 15 16:20:23.553 IST

IS-IS 1 paths to IPv4 Unicast (Level-2) routers
System Id          Metric    Next-Hop           Interface       SNPA
RouterR1             --
RouterR2             **
RouterR3             **
RouterR4             **
RouterR5             **
RouterR6             **

RP/0/RP0/CPU0:RouterR1# show isis topology flex-algo 151 data-plane ip
Mon Nov 15 16:20:30.097 IST

IS-IS 1 paths to IPv4 Unicast (Level-2) routers
System Id          Metric    Next-Hop           Interface       SNPA
RouterR1             --
RouterR2             32        RouterR3             Te0/0/0/14.2    *PtoP*
RouterR3             20        RouterR3             BE1             *PtoP*
RouterR4             30        RouterR3             BE1             *PtoP*
RouterR5             **        RouterR5             Te0/0/0/14.2    *PtoP*
RouterR6             **        RouterR5             Te0/0/0/14.2    *PtoP*
```

These examples verifies the flexible Algorithm 151 IP data-plane IP route.

```
RP/0/RP0/CPU0:RouterR3# show isis route flex-algo 151 2.2.2.51/32 detail
Mon Nov 15 16:01:50.553 IST

L2 2.2.2.51/32 [32/115] Label: None, medium priority
   Installed Nov 15 12:57:39.377 for 03:04:11
     via 32.1.15.2, TenGigE0/0/0/14.2, RouterR4, SRGB Base: 17000, Weight: 0
     src RouterR2.00-00, 2.2.2.2

RP/0/RP0/CPU0:RouterR3# show isis fast-reroute flex-algo 151 2.2.2.51/32 detail
Mon Nov 15 16:02:04.619 IST

L2 2.2.2.51/32 [32/115] Label: None, medium priority
   Installed Nov 15 12:57:39.377 for 03:04:25
     via 32.1.15.2, TenGigE0/0/0/14.2, RouterR4, SRGB Base: 17000, Weight: 0
       Backup path: LFA, via 32.1.13.2, Bundle-Ether1, RouterR2, SRGB Base: 17000, Weight:
 0, Metric: 40
       P: No, TM: 40, LC: No, NP: Yes, D: Yes, SRLG: No
     src RouterR2.00-00, 2.2.2.2

RP/0/RP0/CPU0:RouterR3# show isis ipv6 route flex-algo 151 2:2:2::51/128 detail
Mon Nov 15 16:02:50.749 IST

L2 2:2:2::51/128 [32/115] Label: None, medium priority
   Installed Nov 15 15:45:17.416 for 00:17:33
     via fe80::28a:96ff:fee7:f400, TenGigE0/0/0/14.2, RouterR4, SRGB Base: 17000, Weight:
0
     src RouterR2.00-00, 2:2:2::2

RP/0/RP0/CPU0:RouterR3# show isis ipv6 fast-reroute flex-algo 151 2:2:2::51/128 detail
Mon Nov 15 16:03:02.722 IST

L2 2:2:2::51/128 [32/115] Label: None, medium priority
   Installed Nov 15 15:45:17.416 for 00:17:45
     via fe80::28a:96ff:fee7:f400, TenGigE0/0/0/14.2, RouterR4, SRGB Base: 17000, Weight:
0
       Backup path: LFA, via fe80::2bc:60ff:fe04:74dc, Bundle-Ether1, RouterR2, SRGB Base:
 17000, Weight: 0, Metric: 40
```

```
  P: No, TM: 40, LC: No, NP: Yes, D: Yes, SRLG: No
src RouterR2.00-00, 2:2:2::2
```

Protecting Flexible Algorithm IP Prefixes

Topology-Independent Loop-Free Alternate (TI-LFA) and Microloop avoidance features provide protection to link, node, and Shared Risk Link Groups (SRLG) using Segment Routing.

The IP Flexible Algorithm does not support TI-LFA and Microloop avoidance on its own. However, you can protect the IP Algorithm prefixes in the IP network by enabling Segment Routing in the network. With Segment Routing Flexible Algorithm, you can also protect the IP Flexible Algorithm traffic on its backup path.

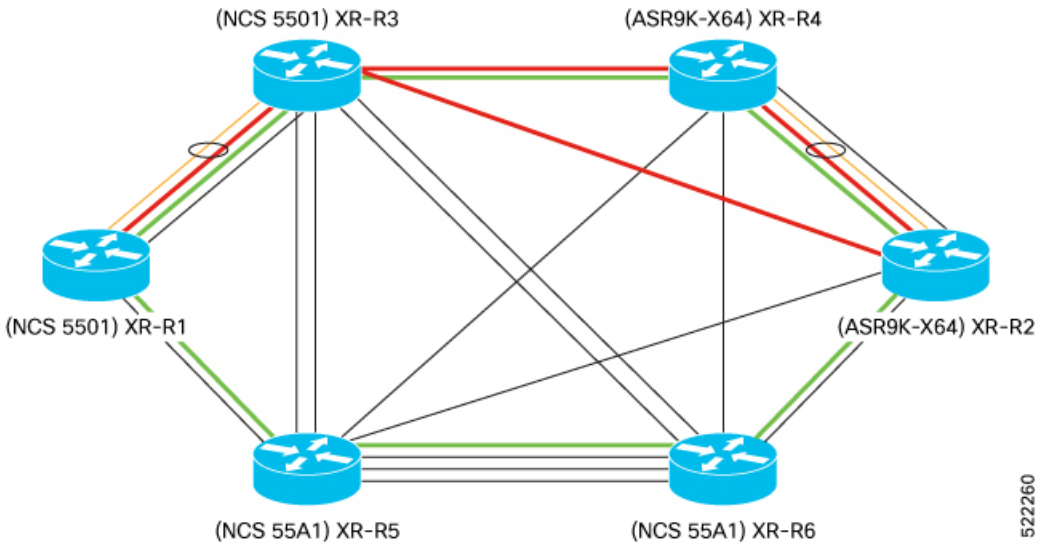
Following conditions are required for TI-LFA protection and/or Microloop avoidance of the IP Flexible Algorithm traffic:

- Both Segment Routing and IP Flexible Algorithms data-planes must be enabled for a particular Flexible Algorithm X in an IS-IS area.
- The exact same set of reachable routers in the IS-IS area is participating in the Segment Routing and IP data-planes for Flexible Algorithm X.

Example: Enabling Flexible Algorithm Protection

The following figure is a six-node topology from R1 to R6 with configurations.

- Node R2 is winning the FAD.
- IP Flexible Algorithm with Flexible Algorithm 151 on nodes R2 ►R4 ► R3 ►R1.
- Segment Routing Flexible Algorithm with Flexible Algorithm 131 on nodes R2 ►R6 ► R5► R1 ► R3 ► R4.
- Both Segment Routing and IP Algorithm with Flexible Algorithm 131 on node R1 ► R3 ►R4 ►R2.



This example shows the Flexible Algorithm 131 and 151 configuration on R2.

```
RP/0/RSP0/CPU0:RouterR2# show run router isis
router isis 1
 is-type level-2-only
 net 49.0000.0000.0000.0012.00
 affinity-map GREEN bit-position 4
flex-algo 131
  data-plane segment-routing ip
  metric-type delay
  advertise-definition
  affinity include-any GREEN
 !
 flex-algo 151
  data-plane ip
  metric-type delay
  advertise-definition
  affinity include-any RED
 !
 address-family ipv4 unicast
  metric-style wide
  segment-routing mpls sr-prefer
  segment-routing mpls unlabeled protection route-policy ip_fa
!
 address-family ipv6 unicast
  metric-style wide
  segment-routing mpls sr-prefer
  segment-routing mpls unlabeled protection route-policy ip_fa

interface Loopback0
  passive
  address-family ipv4 unicast
   prefix-sid index 2
   prefix-sid algorithm 131 index 102
  !
  address-family ipv6 unicast
   prefix-sid index 12
   prefix-sid algorithm 131 index 112
  !
 !

 interface Loopback31
  passive
  address-family ipv4 unicast
  !
  address-family ipv6 unicast
  !
 !
 interface Loopback32
  passive
  address-family ipv4 unicast
  !
  address-family ipv6 unicast
  !
```

This example shows the Loopback interfaces 31 and 51 associated to Flexible Algorithms 131 and 151 respectively.

```
RP/0/RSP0/CPU0:RouterR2# show run int Loopback 31
Mon Nov 15 14:02:03.386 IST
interface Loopback31
 ipv4 address 2.2.2.31 255.255.255.255 algorithm 131
 ipv6 address 2:2:2::31/128 algorithm 131
!
```

```
RP/0/RSP0/CPU0:Router# show run int Loopback 51
Mon Nov 15 14:02:09.832 IST
interface Loopback51
 ipv4 address 2.2.2.51 255.255.255.255 algorithm 151
 ipv6 address 2:2:2::51/128 algorithm 151
!
```

This example shows the 100-Gigabit Ethernet interface associated to the Flexible Algorithms 131 and 151.

```
RP/0/RSP0/CPU0:Router# show run int HundredGigE 0/1/0/0.301
Mon Nov 15 14:02:38.800 IST
interface HundredGigE0/1/0/0.301
 ipv4 address 200.3.1.1 255.255.255.0 algorithm 131
 ipv4 address 200.3.2.1 255.255.255.0 secondary algorithm 151
 ipv6 address 200::3:1:1/112 algorithm 131
 ipv6 address 200::3:2:1/112 algorithm 151
 encapsulation dot1q 301
```

Verification Examples

Use the **show isis flex-algo** command to verify Flexible Algorithm configurations.

This example verifies Flexible Algorithm 131 configuration on both Segment Routing and IP data-planes.

```
RP/0/RSP0/CPU0:RouterR2# show isis flex-algo 131
Mon Nov 15 15:54:01.104 IST

IS-IS 1 Flex-Algo Database

Flex-Algo 131:

  Level-2:
    Definition Priority: 128
    Definition Source: Router.00, (Local)
    Definition Equal to Local: Yes
    Definition Metric Type: Delay
    Definition Flex-Algo Prefix Metric: No
    Exclude Any Affinity Bit Positions:
    Include Any Affinity Bit Positions:  4
    Include All Affinity Bit Positions:
    Exclude SRLGs:
    Disabled: No

  Local Priority: 128
  FRR Disabled: No
  Microloop Avoidance Disabled: No

  Data Plane Segemnt Routing: Yes
  Data Plane IP: Yes
```

This example verifies the Flexible Algorithm 151 configuration on the IP data-plane.

```
RP/0/RSP0/CPU0:RouterR2# show isis flex-algo 151
Mon Nov 15 15:54:06.291 IST

IS-IS 1 Flex-Algo Database

Flex-Algo 151:

  Level-2:
    Definition Priority: 128
    Definition Source: Router.00, (Local)
    Definition Equal to Local: Yes
    Definition Metric Type: Delay
```

```
    Definition Flex-Algo Prefix Metric: No
    Exclude Any Affinity Bit Positions:
    Include Any Affinity Bit Positions:  4
    Include All Affinity Bit Positions:
    Exclude SRLGs:
    Disabled: No

  Local Priority: 128
  FRR Disabled: No
  Microloop Avoidance Disabled: No

  Data Plane Segemnt Routing: No
  Data Plane IP: Yes
```

This example verifies the Loopback interface 32 configuration.

```
RP/0/RSP0/CPU0:RouterR2#show isis interface Loopback 31          VERIFY ALGO AND IP IN ISIS
 INTERFACE
Mon Nov 15 16:42:30.819 IST

Loopback31                 Enabled
  Adjacency Formation:     Disabled (Passive in IS-IS cfg)
  Prefix Advertisement:    Enabled
 !

  IPv4 Address Family:     Enabled
    Protocol State:        Up
    Forwarding Address(es): Unknown (Intf passive in IS-IS cfg)
    Global Prefix(es):     2.2.2.31/32 (131)
  IPv6 Address Family:     Enabled
    Protocol State:        Up
    Forwarding Address(es): Unknown (Intf passive in IS-IS cfg)
    Global Prefix(es):     2:2:2::31/128 (131)
!
```

This example verifies the IP Algo Prefix Advertisements in ISIS database on R2.

```
RP/0/RP0/CPU0:RouterR1# show isis database detail verbose RouterR2
Mon Nov 15 16:05:22.188 IST

IS-IS 1 (Level-2) Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime/Rcvd  ATT/P/OL
RouterR2.00-00          0x0000004c   0xd97e        773  /1200         0/0/0
  Area Address:   49.0000
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 2.2.2.31/32 D:0 Metric: 0 Algorithm: 131
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
    Source Router ID: 2.2.2.2
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 2.2.2.51/32 D:0 Metric: 0 Algorithm: 151
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
    Source Router ID: 2.2.2.2
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 200.3.1.0/24 D:0 Metric: 10 Algorithm: 131

    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 200.3.2.0/24 D:0 Metric: 10 Algorithm: 151

    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv6 Algo Prefix: MT (IPv6 Unicast) 2:2:2::31/128 D:0 Metric: 0 Algorithm: 131
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv6 Algo Prefix: MT (IPv6 Unicast) 2:2:2::51/128 D:0 Metric: 0 Algorithm: 151
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  NLPID:          0xcc
  NLPID:          0x8e
  IP Address:     2.2.2.2
  Router ID:      2.2.2.2
  Metric: 0          IP-Extended 2.2.2.2/32
```

```
    Prefix-SID Index: 2, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix-SID Index: 102, Algorithm:131, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix Attribute Flags: X:0 R:0 N:1 E:0 A:0
    Source Router ID: 2.2.2.2
  Metric: 0          MT (IPv6 Unicast) IPv6 2:2:2::2/128
    Prefix-SID Index: 12, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix-SID Index: 112, Algorithm:131, R:0 N:1 P:0 E:0 V:0 L:0
```

This example verifies the Flexible Algorithm participation in Segment Routing and IP data-plane in IS-IS database on R2.

```
RP/0/RP0/CPU0:RouterR1# show isis database detail verbose RouterR2
Mon Nov 15 16:05:22.188 IST

Router Cap:     2.2.2.2 D:0 S:0
    Segment Routing: I:1 V:1, SRGB Base: 17000 Range: 7000
    SR Local Block: Base: 15000 Range: 1000
    Node Maximum SID Depth:
      Label Imposition: 10
    SR Algorithm:
      Algorithm: 0
      Algorithm: 131
    IP Algorithm:
      Algorithm: 131
      Algorithm: 151
    Flex-Algo Definition:
      Algorithm: 131 Metric-Type: 1 Alg-type: 0 Priority: 128
      Flex-Algo Include-Any Ext Admin Group:
        0x00000010
    Flex-Algo Definition:
      Algorithm: 151 Metric-Type: 1 Alg-type: 0 Priority: 128
      Flex-Algo Include-Any Ext Admin Group:
        0x00000010
```

This exanple verifies Flexible Algorithm 131 is present in both topologies—Segment Routing and IP data-plane

```
RP/0/RP0/CPU0:RouterR1# show isis topology flex-algo 131 data-plane segment-routing
Mon Nov 15 16:20:05.271 IST

IS-IS 1 paths to IPv4 Unicast (Level-2) routers
System Id          Metric    Next-Hop           Interface       SNPA
RouterR1             --
RouterR2             32        RouterR5             Te0/0/0/14.2    *PtoP*
RouterR3             20        RouterR3             BE1             *PtoP*
RouterR4             30        RouterR3             BE1             *PtoP*
RouterR5             12        RouterR5             Te0/0/0/14.2    *PtoP*
RouterR6             22        RouterR5             Te0/0/0/14.2    *PtoP*

RP/0/RP0/CPU0:RouterR1# show isis topology flex-algo 131 data-plane ip
Mon Nov 15 16:20:14.015 IST

IS-IS 1 paths to IPv4 Unicast (Level-2) routers
System Id          Metric    Next-Hop           Interface       SNPA
RouterR1             --
RouterR2             32        RouterR5             Te0/0/0/14.2    *PtoP*
RouterR3             20        RouterR3             BE1             *PtoP*
RouterR4             30        RouterR3             BE1             *PtoP*
RouterR5             12        RouterR5             Te0/0/0/14.2    *PtoP*
RouterR6             22        RouterR5             Te0/0/0/14.2    *PtoP*
```

This example shows Segment Routing Flexible Algorithm 131 prefix with Prefix SIDs.

```
RP/0/RSP0/CPU0:RouterR2# show isis route flex-algo 131 1.1.1.1/32 detail            SR FA
131 PREFIX WITH PREFIX-SID
Mon Nov 15 15:32:53.170 IST
```

```
L2 1.1.1.1/32 [31/115] Label: 17101, medium priority
   Installed Nov 15 13:46:40.902 for 01:46:13
     via 32.1.24.2, TenGigE0/1/0/3/6, Label: 17101, RouterR4, SRGB Base: 17000, Weight: 0
     src RouterR1.00-00, 1.1.1.1, prefix-SID index 101, R:0 N:1 P:0 E:0 V:0 L:0, Alg:131
RP/0/RSP0/CPU0:Router#
RP/0/RSP0/CPU0:Router2# show isis fast-reroute flex-algo 131 1.1.1.1/32 detail
Mon Nov 15 15:32:58.794 IST

L2 1.1.1.1/32 [31/115] Label: 17101, medium priority
   Installed Nov 15 13:46:40.902 for 01:46:19
     via 32.1.24.2, TenGigE0/1/0/3/6, Label: 17101, RouterR4, SRGB Base: 17000, Weight: 0
       Backup path: TI-LFA (link), via 32.1.26.2, TenGigE0/1/0/3/5.2 tb5-r6, SRGB Base:
17000, Weight: 0, Metric: 120
         P node: RouterR6.00 [6.6.6.6], Label: ImpNull
         Q node: RouterR5.00 [5.5.5.5], Label: 25009
         Prefix label: 17101
         Backup-src: RouterR1.00
       P: No, TM: 120, LC: No, NP: No, D: No, SRLG: No
     src RouterR1.00-00, 1.1.1.1, prefix-SID index 101, R:0 N:1 P:0 E:0 V:0 L:0, Alg:131
```

This example shows IP Flexible Algorithm 131 prefix with IP Prefixes.

```
RP/0/RSP0/CPU0:RouterR2#show isis route flex-algo 131 1.1.1.31/32 detail
Mon Nov 15 15:33:15.970 IST

L2 1.1.1.31/32 [31/115] Label: None, medium priority
   Installed Nov 15 13:46:34.923 for 01:46:42
     via 32.1.24.2, TenGigE0/1/0/3/6, tb5-r4, SRGB Base: 17000, Weight: 0
     src RouterR1.00-00, 1.1.1.1


!!TI-LFA backup is seen only if both IP AND SR TOPOLOGY ARE CONGRUENT

RP/0/RSP0/CPU0:RouterR2#show isis fast-reroute flex-algo 131 1.1.1.31/32 detail
Mon Nov 15 15:33:27.404 IST

L2 1.1.1.31/32 [31/115] Label: None, medium priority
   Installed Nov 15 13:46:34.923 for 01:46:54
     via 32.1.24.2, TenGigE0/1/0/3/6, tb5-r4, SRGB Base: 17000, Weight: 0
       Backup path: TI-LFA (link), via 32.1.26.2, TenGigE0/1/0/3/5.2 tb5-r6, SRGB Base:
17000, Weight: 0, Metric: 120
         P node: tb5-r6.00 [6.6.6.6], Label: ImpNull
         Q node: tb5-r5.00 [5.5.5.5], Label: 25009
         Prefix label: None
         Backup-src: RouterR1.00
       P: No, TM: 120, LC: No, NP: No, D: No, SRLG: No
     src RouterR1.00-00, 1.1.1.1
```

Flexible-Algorithm Redistribution in IP Networks

Table 14: Feature History Table

Feature Name	Release Information	Feature Description

Earlier, there was no way to redistribute routers into a particular Flex-Algorithm from another domain. Due to this, when you were running IP Flex-Algorithms in a domain and you were redistributing Flex-Algorithms prefixes from another domain, all prefixes would go to base algorithm (algorithm 0). For more information

on Flex-Algorithm and the algorithm numbers, see *IGP Flexible Algorithm in IP Networks* in the *Routing Configuration Guide for Cisco NCS 5500 Series Routers*.

Starting from Release 7.9.1, you can now redistribute prefixes into a particular Flex-Algorithm between 128-255. This allows Link State IGPs (OSPF, ISIS) to compute paths using various constraints. You can have more control over the traffic that moves between networks using various routing protocols by configuring Flex-Algorithms for prefixes that are redistributed into IS-IS protocols.

This feature allows you to:

- Set an algorithm in the prefixes
- Match the prefixes based on the algorithm

Set an Algorithm

The set algorithm is added in the route policy mechanism. By using this **set algorithm** command, you can set the Flex-Algorithm for the prefixes that are redistributed into IS-IS protocols. The prefix advertisement after route redistribution can be altered via **set algorithm** in the routing policy.

Limitation:

- The Route Policy accepts algorithm number from 0 to 255.
- The IS-IS protocol only handles algorithm 0, 128-255.
- The algorithm number 1 through 127 in IS-IS protocol are misconfigured algorithms and treated as algorithm 0. Due to this, if you are redistributing Flex-Algorithms prefixes (1 through 127) from another domain, all prefixes would go to base algorithm (algorithm 0).

Use Set Algorithm in RPL

Configuration Example

1. Set the prefixes (that are redistributed into IS-IS protocol) into Flex-Algorithm 128 using the route-policy.

```
Router(config)#prefix-set PFX_ALGO128
Router(config-pfx)#44.44.44.128/32,
Router(config-pfx)#44:44:44::128/128
Router(config-pfx)#end-set
```

2. Define the route-policy using set algorithm to set Flex-Algorithm 128 for prefix-set *PFX_ALGO128*.

```
Router(config)#route-policy BGP_TO_ISIS
Router(config-rpl)# if destination in PFX_ALGO128 then
Router(config-rpl-if)# set tag 200
Router(config-rpl-if)# set algorithm 128
Router(config-rpl-if)# pass
Router(config-rpl-if)# else
Router(config-rpl-else)# drop
Router(config-rpl-else)# endif
Router(config-rpl)#end-policy
Router(config)#commit
```

3. Use the route-policy while redistributing Flex-Algorithm 128 into IS-IS protocol.

```
Router(config)#router isis 100
Router(config-isis)#address-family ipv4 unicast
```

```
Router(config-isis-af)#redistribute bgp 100 route-policy BGP_TO_ISIS metric-type
rib-metric-as-external
Router(config-isis-af)#exit
Router(config-isis)#address-family ipv6 unicast
Router(config-isis-af)#redistribute bgp 100 route-policy BGP_TO_ISIS metric-type
rib-metric-as-external
Router(config-isis-af)#commit
Router(config-isis-af)#end
```

Running Configuration

```
prefix-set PFX_ALGO128
  44.44.44.128/32,
  44:44:44::128/128
end-set
!
route-policy BGP_TO_ISIS
  if destination in PFX_ALGO128 then
    set tag 200
    set algorithm 128
    pass
  else
    drop
  endif
end-policy

router isis 100
 address-family ipv4 unicast
  metric-style wide
  redistribute bgp 100 route-policy BGP_TO_ISIS metric-type rib-metric-as-external
  segment-routing mpls
 !

router isis 100
 address-family ipv6 unicast
  metric-style wide
  redistribute bgp 100 route-policy BGP_TO_ISIS metric-type rib-metric-as-external
  segment-routing mpls
```

Verification

Run the **show isis database** command in IS-IS protocol database to check whether RPL is applied and redistributed prefixes has Flex-Algorithm 128 attached.

```
Router#show isis instance 100 database R4.00-01 verbose | begin 44.44.44.128/32
Tue Dec 13 09:12:41.395 UTC
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 44.44.44.128/32 D:0 Metric: 2 Algorithm:
128
    Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
      algo 128 due to RPL with set algo.
    Admin. Tag: 200
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 44.44.44.128/32 D:0 Metric: 10 Algorithm:
 129
    Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
    Admin. Tag: 200
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 44.44.44.128/32 D:0 Metric: 10 Algorithm:
 130
    Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
    Admin. Tag: 200
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 44.44.44.128/32 D:0 Metric: 10 Algorithm:
```

```
131
   Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
   Admin. Tag: 200
 IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 44.44.44.128/32 D:0 Metric: 10 Algorithm:
132
   Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
   Admin. Tag: 200
 Metric: 0          MT (IPv6 Unicast) IPv6-External 6:6:6:6::6/128
   Admin. Tag: 200
   Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
 Metric: 10         MT (IPv6 Unicast) IPv6-External 7:7:7:7::7/128
   Admin. Tag: 200
   Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
 IPv6 Algo Prefix: MT (IPv6 Unicast) 44:44:44::132/128 D:0 Metric: 0 Algorithm: 132
   Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
 IPv6 Algo Prefix: MT (IPv6 Unicast) 66:66:66::128/128 D:0 Metric: 0 Algorithm: 128
   Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
   Admin. Tag: 200
 IPv6 Algo Prefix: MT (IPv6 Unicast) 44:44:44::128/128 D:0 Metric: 2 Algorithm: 128
   Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
   Admin. Tag: 200
 IPv6 Algo Prefix: MT (IPv6 Unicast) 44:44:44::129/128 D:0 Metric: 10 Algorithm: 129
   Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
   Admin. Tag: 200
```

Match an Algorithm

You can configure the match algorithms in RPL and use them to filter out based on algorithms during redistribution. This mechanism acts as a filter based on the algorithm number in the prefixes while redistributing.

Use Match Algorithm in RPL

You can use the match algorithm as a conditional expression to select or filter the prefix matching algorithm during route redistribution.

Configuration Example

1. Set a route-policy that is used to filter out Flex-Algorithm 128 and algorithm 0 prefixes.

```
Router(config)#route-policy ISIS_TO_BGP
Router(config-rpl)#  if algorithm is 128 or algorithm is 0 then
Router(config-rpl-if)# pass
Router(config-rpl-if)# else
Router(config-rpl-else)# drop
Router(config-rpl-else)# endif
Router(config-rpl)#end-policy
Router(config)#commit
```

2. Apply the route-policy to filter out Flex-Algorithm 128 and algorithm 0 prefixes while redistributing IS-IS protocol into BGP.

```
Router(config)#router bgp 100
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#  redistribute isis 100 route-policy ISIS_TO_BGP
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)# redistribute isis 100 route-policy ISIS_TO_BGP
Router(config-bgp-af)#commit
```

Running Configuration

```
route-policy ISIS_TO_BGP
  if algorithm is 128 or algorithm is 0 then
    pass
  else
    drop
  endif
end-policy
!

router bgp 100
 address-family ipv4 unicast
  redistribute isis 100 route-policy ISIS_TO_BGP
 !

router bgp 100
 address-family ipv6 unicast
  redistribute isis 100 route-policy ISIS_TO_BGP
 !
```

Verification

The **show isis route flex-algo <*algorithm number*>** command displays the routes available into Flex-Algorithm 128:

```
Router#show isis route flex-algo 128
Tue Dec 13 09:34:09.502 UTC
IS-IS 100 IPv4 Unicast routes Flex-Algo 128
Codes: L1 - level 1, L2 - level 2, ia - interarea (leaked into level 1)
       df - level 1 default (closest attached router), su - summary null
       C - connected, S - static, R - RIP, B - BGP, O - OSPF
       E - EIGRP, A - access/subscriber, M - mobile, a - application
       i - IS-IS (redistributed from another instance)
Maximum parallel path count: 8
L2 11.11.11.128/32 [20/115]
     via 2.4.0.2, HundredGigE0/0/0/1, R2, SRGB Base: 16000, Weight: 0
L2 22.22.22.128/32 [10/115]
     via 2.4.0.2, HundredGigE0/0/0/1, R2, SRGB Base: 16000, Weight: 0
L2 33.33.33.128/32 [30/115]
     via 2.4.0.2, HundredGigE0/0/0/1, R2, SRGB Base: 16000, Weight: 0
C  44.44.44.128/32
     is directly connected, Loopback129
L2 55.55.55.128/32 [40/115]
     via 2.4.0.2, HundredGigE0/0/0/1, R2, SRGB Base: 16000, Weight: 0
```

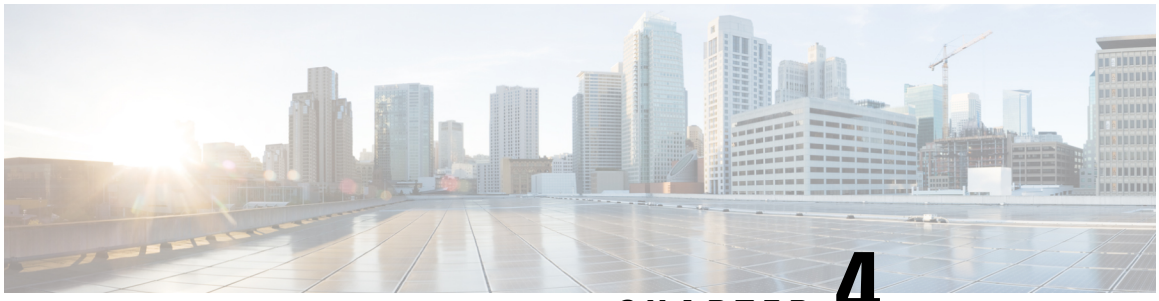
The **show route *prefix* detail** command displays the prefix `11.11.11.128/32` prefix belongs to Flex-Algorithm 128.

```
Router#show route 11.11.11.128/32 detail
Tue Dec 13 09:35:16.681 UTC
Routing entry for 11.11.11.128/32
  Known via "isis 100", distance 115, metric 4 (algo 128), type level-2
  Installed Dec 13 07:56:46.972 for 01:38:29
  Routing Descriptor Blocks
    4.5.0.5, from 1.1.1.1, via HundredGigE0/0/0/2, Backup (Local-LFA)
      Route metric is 54
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
```

```
    Path id:65       Path ref count:1
    NHID:0x3(Ref:29)
  2.4.0.2, from 1.1.1.1, via HundredGigE0/0/0/1, Protected
    Route metric is 4
    Label: None
    Tunnel ID: None
    Binding Label: None
    Extended communities count: 0
    Path id:1       Path ref count:0
    NHID:0x2(Ref:32)
    Backup path id:65
Route version is 0x2 (2)
No local label
IP Precedence: Not Set
QoS Group ID: Not Set
Flow-tag: Not Set
Fwd-class: Not Set
Route Priority: RIB_PRIORITY_NON_RECURSIVE_MEDIUM (7) SVD Type RIB_SVD_TYPE_LOCAL
Download Priority 1, Download Version 170
No advertising protos.
```

Use the **show bpg ipv4 unicast advertised summary** command to verify RPL filtering.

```
Router#show bgp ipv4 unicast advertised summary
Tue Dec 13 09:37:53.596 UTC
Network            Next Hop        From            Advertised to
1.1.1.1/32         4.6.0.4         Local           4.6.0.6
1.2.0.0/24         4.6.0.4         Local           4.6.0.6
1.3.0.0/24         4.6.0.4         Local           4.6.0.6
2.2.2.2/32         4.6.0.4         Local           4.6.0.6
2.4.0.0/24         4.6.0.4         Local           4.6.0.6
3.3.3.3/32         4.6.0.4         Local           4.6.0.6
3.5.0.0/24         4.6.0.4         Local           4.6.0.6
4.4.4.4/32         4.6.0.4         Local           4.6.0.6
4.5.0.0/24         4.6.0.4         Local           4.6.0.6
4.6.0.0/24         4.6.0.4         Local           4.6.0.6
5.5.5.5/32         4.6.0.4         Local           4.6.0.6
5.6.0.0/24         4.6.0.4         Local           4.6.0.6
11.11.11.128/32    4.6.0.4         Local           4.6.0.6
22.22.22.128/32    4.6.0.4         Local           4.6.0.6
33.33.33.128/32    4.6.0.4         Local           4.6.0.6
44.44.44.128/32    4.6.0.4         Local           4.6.0.6
55.55.55.128/32    4.6.0.4         Local           4.6.0.6
Processed 17 prefixes, 17 paths
```



CHAPTER 4

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.

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.

This module describes how to implement and monitor RIB on your network.

- Verify RIB Configuration Using Routing Table, on page 163
- Verify Networking and Routing Problems, on page 164
- Disable RIB Next-hop Dampening, on page 166
- Enable RCC and LCC On-demand Scan, on page 167
- Enable RCC and LCC Background Scan, on page 168
- References for RIB, on page 169

Verify RIB Configuration Using 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.

Procedure

Step 1 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **summary** [**detail**] [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route summary
```

Displays route summary information about the specified routing table.

- The default table summarized is the IPv4 unicast routing table.

Step 2 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] [*protocol* [*instance*] | *ip-address mask*] [**standby**] [**detail**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast
```

Displays more detailed route information about the specified routing table.

- This command is usually issued with an IP address or other optional filters to limit its display. Otherwise, it displays all routes from the default IPv4 unicast routing table, which can result in an extensive list, depending on the configuration of the network.

Output of show route best-local 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
        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, HundredGigE 4/0/0/10/0/1/0
                Backup  O E2 [110/1] via 10.12.12.2, HundredGigE 3/0/0/10/0/1/0
```

Verify Networking and Routing Problems

Perform this task to verify the operation of routes between nodes.

Procedure

Step 1 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] [*protocol* [*instance*] | *ip-address mask*] [**standby**] [**detail**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast 192.168.1.11/8
```

Displays the current routes in RIB.

Step 2 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **backup** [*ip-address*] [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast backup 192.168.1.11/8
```

Displays backup routes in RIB.

Step 3 **show route** [**vrf** { *vrf-name* | **all** }] [**ipv4** | **ipv6**] [**unicast** | **safi-all**] **best-local** *ip-address* [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast best-local 192.168.1.11
```

Displays the best-local address to use for return packets from the given destination.

Step 4 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **connected** [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast connected
```

Displays the current connected routes of the routing table.

Step 5 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **local** [*interface*] [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast local
```

Displays local routes for receive entries in the routing table.

Step 6 **show route** [**vrf** { *vrf-name* | **all** }] [**ipv4** | **ipv6**] [**unicast** | **safi-all**] **longer-prefixes** { *ip-address mask* | *ip-address* / *prefix-length* } [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast longer-prefixes 192.168.1.11/8
```

Displays the current routes in RIB that share a given number of bits with a given network.

Step 7 **show route** [**vrf** { *vrf-name* | **all** }] [**ipv4** | **ipv6**] [**unicast** | **safi-all**] **next-hop** *ip-address* [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast next-hop 192.168.1.34
```

Displays the next-hop gateway or host to a destination address.

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
```

```
        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
```

Disable RIB Next-hop Dampening

Perform this task to disable RIB next-hop dampening.

Procedure

Step 1 **router rib**

Example:

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

Enters RIB configuration mode.

Step 2 **address-family** { **ipv4** | **ipv6** } **next-hop dampening disable**

Example:

```
RP/0/RP0/CPU0:router(config-rib)# address-family ipv4 next-hop dampening disable
```

Disables next-hop dampening for IPv4 address families.

Step 3 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

Output of show route next-hop Command: Example

The following is sample output from the **show route resolving-next-hop** command:

```
  show route resolving-next-hop 10.0.0.1

  Nexthop matches 0.0.0.0/0
    Known via "static", distance 200, metric 0, candidate default path
    Installed Aug 18 00:59:04.448
    Directly connected nexthops
      172.29.52.1, via MgmtEth0/

/CPU0/0
        Route metric is 0
      172.29.52.1, via MgmtEth0/RP1/CPU0/0
        Route metric is 0
```

Enable 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.

Procedure

Step 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*]

Example:

```
RP/0/RP0/CPU0:router#show rcc ipv6 unicast 2001:DB8::/32 vrf vrf_1
```

Or

```
RP/0/RP0/CPU0:router#show lcc ipv6 unicast 2001:DB8::/32 vrf vrf_1
```

Runs on-demand Route Consistency Checker (RCC) or Label Consistency Checker (LCC).

Step 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**

Example:

```
RP/0/RP0/CPU0:router#clear rcc ipv6 unicast log
```

Or

```
RP/0/RP0/CPU0:router#show lcc ipv6 unicast log
```

Clears the log of previous scans.

Enable RCC and LCC Background Scan

Perform this task to run a background scan for Route Consistency Checker (RCC) and Label Consistency Checker (LCC).

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 Use one of these commands:

- **rcc** {**ipv4** | **ipv6**} **unicast** {**enable** | **period** *milliseconds*}
- **lcc** {**ipv4** | **ipv6**} **unicast** {**enable** | **period** *milliseconds*}

Example:

```
RP/0/RP0/CPU0:router(config)#rcc ipv6 unicast enable

RP/0/RP0/CPU0:router(config)#rcc ipv6 unicast period 500
```

Or

```
RP/0/RP0/CPU0:router(config)#lcc ipv6 unicast enable

RP/0/RP0/CPU0:router(config)#lcc ipv6 unicast period 500
```

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).

Step 3 Use the **commit** or **end** command.

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

end —Prompts user to take one of these actions:

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

Step 4 Use one of these commands.

- **show rcc** {**ipv4**| **ipv6**} **unicast** [**summary** | **scan-id** *scan-id-value*]
- **show lcc** {**ipv4**| **ipv6**} **unicast** [**summary** | **scan-id** *scan-id-value*]

Example:

```
RP/0/RP0/CPU0:router#show rcc ipv6 unicast statistics scan-id 120
```

Or

```
RP/0/RP0/CPU0:router#show lcc ipv6 unicast statistics scan-id 120
```

Displays statistics about background scans.

- **summary**—Displays the current ongoing scan id and a summary of the previous few scans.
- **scan-id** *scan-id-value*—Displays details about a specific scan.

Enabling RCC and LCC: 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
```

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
```

References for RIB

This section provides additional conceptual information on RIB. It includes the following topics:

- RIB Data Structures in BGP and Other Protocols, on page 170
- RIB Administrative Distance, on page 170
- RIB Statistics, on page 171
- RIB Quarantining, on page 171
- Route and Label Consistency Checker, on page 172

RIB Data Structures in BGP and Other Protocols

RIB uses processes and maintains data structures distinct from other routing applications, such as Border Gateway Protocol (BGP) and other unicast routing protocols. However, these routing protocols use internal data structures similar to what RIB uses, and may internally refer to the data structures as a RIB. For example, BGP routes are stored in the BGP RIB (BRIB). RIB processes are not responsible for the BRIB, which are handled by BGP.

The table used by the line cards and RP to forward packets is called the Forwarding Information Base (FIB). RIB processes do not build the FIBs. Instead, RIB downloads the set of selected best routes to the FIB processes, by the Bulk Content Downloader (BCDL) process, onto each line card. FIBs are then constructed.

RIB Administrative Distance

Forwarding is done based on the longest prefix match. If you are forwarding a packet destined to 10.0.2.1, you prefer 10.0.2.0/24 over 10.0.0.0/16 because the mask /24 is longer (and more specific) than a /16. Routes from different protocols that have the same prefix and length are chosen based on administrative distance. For instance, the Open Shortest Path First (OSPF) protocol has an administrative distance of 110, and the Intermediate System-to-Intermediate System (IS-IS) protocol has an administrative distance of 115. If IS-IS and OSPF both download 10.0.1.0/24 to RIB, RIB would prefer the OSPF route because OSPF has a lower administrative distance. Administrative distance is used only to choose between multiple routes of the same length.

This table lists default administrative distances for the common protocols.

Table 15: Default Administrative Distances

Protocol	Administrative Distance Default
Connected or local routes	0
Static routes	1
External BGP routes	20
OSPF routes	110
IS-IS routes	115
Internal BGP routes	200

The administrative distance for some routing protocols (for instance IS-IS, OSPF, and BGP) can be changed. See the protocol-specific documentation for the proper method to change the administrative distance of that protocol.

Note Changing the administrative distance of a protocol on some but not all routers can lead to routing loops and other undesirable behavior. Doing so is not recommended.

RIB 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.

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 using the appropriate command modes 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.

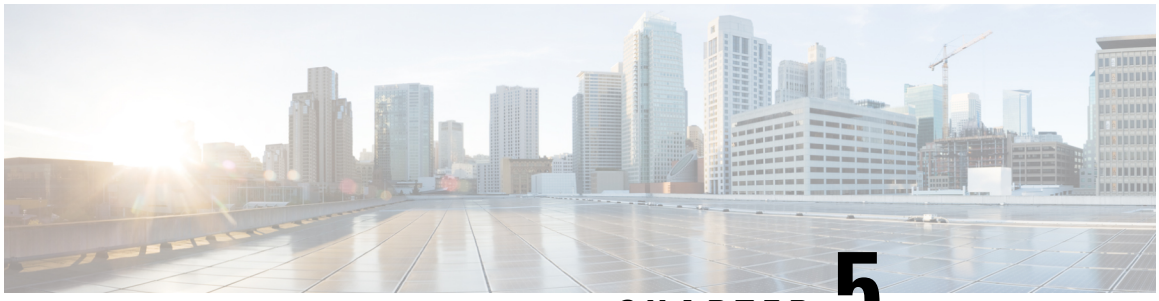
RCC runs on the Route Processor. FIB checks for errors on the line card and forwards first the 20 error reports to RCC. RCC receives error reports from all nodes, summarizes them (checks for exact match), and adds it to two queues, soft or hard. Each queue has a limit of 1000 error reports and there 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.



CHAPTER 5

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.

- Restrictions for Implementing Routing Policy, on page 173
- Define Route Policy, on page 174
- Attach Routing Policy to BGP Neighbor, on page 176
- Modify Routing Policy Using Text Editor, on page 177
- References for Routing Policy, on page 180

Restrictions for Implementing Routing Policy

These restrictions apply when working with Routing Policy Language implementation:

- Border Gateway Protocol (BGP), integrated Intermediate System-to-Intermediate System (IS-IS), or Open Shortest Path First (OSPF) must be configured in your network.
- An individual policy definition of up to 1000 statements are supported. The total number of statements within a policy can be extended to 4000 statements using hierarchical policy constructs. However, this limit is restricted with the use of **apply** statements.
- When a policy that is attached directly or indirectly to an attach point needs to be modified, a single **commit** operation cannot be performed when:
  - Removing a set or policy referred by another policy that is attached to any attach point directly or indirectly.
  - Modifying the policy to remove the reference to the same set or policy that is getting removed.

  The **commit** must be performed in two steps:

1. Modify the policy to remove the reference to the policy or set and then **commit**.
2. Remove the policy or set and **commit**.

- Per-vrf label mode is not supported for Carrier Supporting Carrier (CSC) network with internal and external BGP multipath setup.
- You cannot change the next hop address to an IPv6 address through RPL policy for a route that starts from an IPv4 peer.
- Starting with Cisco IOS XR Release 7.10.2, the CLI no longer accepts IPv4 addresses with leading zeros. Use canonical dotted-decimal IPv4 addresses such as 198.51.100.1.

  If your existing configuration, scripts, or Route Policy Language (RPL) expressions contain IPv4 addresses with leading zeros, NETCONF and YANG reject the address. This can cause configuration retrieval, synchronization, or BGP-related configuration to fail. Therefore, configure all IPv4 addresses in the correct format, ensuring that no part of the address includes leading zeros.

Define Route Policy

This task explains how to define a route policy.

Note

- If you want to modify an existing routing policy using the command-line interface (CLI), you must redefine the policy by completing this task.
- Modifying the RPL scale configuration may take a long time.
- BGP may crash either due to large scale RPL configuration changes, or during consecutive RPL changes. To avoid BGP crash, wait until there are no messages in the BGP In/Out queue before committing further changes.

Tip You can programmatically configure the route policy using `openconfig-routing-policy.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide* .

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **route-policy** *name* [*parameter1* , *parameter2* , . . . , *parameterN*]

Example:

```
RP/0/RP0/CPU0:router(config)# route-policy sample1
```

Enters route-policy configuration mode.

- After the route-policy has been entered, a group of commands can be entered to define the route-policy.

Step 3 **end-policy**

Example:

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

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

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

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

end —Prompts user to take one of these actions:

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

Routing Policy Definition: Example

In the following example, a BGP route policy named sample1 is defined using the **route-policy** *name* command. The policy compares the network layer reachability information (NLRI) to the elements in the prefix set test. If it evaluates to true, the policy performs the operations in the *then* clause. If it evaluates to false, the policy performs the operations in the *else* clause, that is, sets the MED value to 200 and adds the community 2:100 to the route. The final steps of the example commit the configuration to the router, exit configuration mode, and display the contents of route policy sample1.

```
configure
 route-policy sample1
  if destination in test then
   drop
  else
   set med 200
   set community (2:100) additive
  endif
 end-policy
 end
show config running route-policy sample1
Building configuration...
 route-policy sample1
  if destination in test then
   drop
  else
   set med 200
   set community (2:100) additive
  endif
 end-policy
```

Attach Routing Policy to 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.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

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

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 125
```

Configures a BGP routing process and enters router configuration mode.

- The *as-number* argument identifies the autonomous system in which the router resides. Valid values are from 0 to 65535. Private autonomous system numbers that can be used in internal networks range from 64512 to 65535.

Step 3 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.20
```

Specifies a neighbor IP address.

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

Example:

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

Specifies the address family.

Step 5 **route-policy** *policy-name* { **in** | **out** }

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy example1 in
```

Attaches the route-policy, which must be well formed and predefined.

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

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

end —Prompts user to take one of these actions:

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

Modify Routing Policy Using Text Editor

This task explains how to modify an existing routing policy using a text editor.

Procedure

Step 1 **edit** { **route-policy** | **prefix-set** | **as-path-set** | **community-set** | **extcommunity-set** { **rt** | **soo** } | **policy-global** | **rd-set** } *name* [**nano** | **emacs** | **vim** | **inline** { **add** | **prepend** | **remove** } *set-element*]

Example:

```
RP/0/RP0/CPU0:router# edit route-policy sample1
```

Identifies the route policy, prefix set, AS path set, community set, or extended community set name to be modified.

- A copy of the route policy, prefix set, AS path set, community set, or extended community set is copied to a temporary file and the editor is launched.
- After editing with Nano, save the editor buffer and exit the editor by using the Ctrl-X keystroke.
- After editing with Emacs, save the editor buffer by using the Ctrl-X and Ctrl-S keystrokes. To save and exit the editor, use the Ctrl-X and Ctrl-C keystrokes.
- After editing with Vim, to write to a current file and exit, use the :wq or :x or ZZ keystrokes. To quit and confirm, use the :q keystrokes. To quit and discard changes, use the :q! keystrokes.

Step 2 **show rpl route-policy** [*name* [**detail**] | **states** | **brief**]

Example:

```
RP/0/RP0/CPU0:router# show rpl route-policy sample2
```

(Optional) Displays the configuration of a specific named route policy.

- Use the **detail** keyword to display all policies and sets that a policy uses.
- Use the **states** keyword to display all unused, inactive, and active states.

- Use the **brief** keyword to list the names of all extended community sets without their configurations.

Step 3 **show rpl prefix-set** [*name* | **states** | **brief**]

Example:

```
RP/0/RP0/CPU0:router# show rpl prefix-set prefixset1
```

(Optional) Displays the contents of a named prefix set.

- To display the contents of a named AS path set, community set, or extended community set, replace the **prefix-set** keyword with **as-path-set** , **community-set** , or **extcommunity-set** , respectively.

Simple Inbound Policy: Example

The following policy discards any route whose network layer reachability information (NLRI) specifies a prefix longer than /24, and any route whose NLRI specifies a destination in the address space reserved by RFC 1918. For all remaining routes, it sets the MED and local preference, and adds a community to the list in the route.

For routes whose community lists include any values in the range from 101:202 to 106:202 that have a 16-bit tag portion containing the value 202, the policy prepends autonomous system number 2 twice, and adds the community 2:666 to the list in the route. Of these routes, if the MED is either 666 or 225, then the policy sets the origin of the route to incomplete, and otherwise sets the origin to IGP.

For routes whose community lists do not include any of the values in the range from 101:202 to 106:202, the policy adds the community 2:999 to the list in the route.

```
prefix-set too-specific
 0.0.0.0/0 ge 25 le 32
end-set

prefix-set rfc1918
 10.0.0.0/8 le 32,
 172.16.0.0/12 le 32,
 192.168.0.0/16 le 32
end-set

route-policy inbound-tx
 if destination in too-specific or destination in rfc1918 then
  drop
 endif
 set med 1000
 set local-preference 90
 set community (2:1001) additive
 if community matches-any ([101..106]:202) then
  prepend as-path 2.30 2
  set community (2:666) additive
 if med is 666 or med is 225 then
  set origin incomplete
 else
  set origin igp
 endif
else
  set community (2:999) additive
```

```
endif
end-policy

router bgp 2
 neighbor 10.0.1.2 address-family ipv4 unicast route-policy inbound-tx in
```

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.

```
prefix-set bogon
  10.0.0.0/8 ge 8 le 32,
  0.0.0.0,
  0.0.0.0/0 ge 27 le 32,
  192.168.0.0/16 ge 16 le 32
end-set
!
route-policy in-100
  apply common-inbound
  if community matches-any ([100..120]:135) then
    apply set-lpref-prepend (100,100,2)
    set community (2:1234) additive
  else
    set local-preference 110
  endif
  if community matches-any ([100..666]:[100..999]) then
    set med 444
    set local-preference 200
    set community (no-export) additive
  endif
end-policy
!
route-policy in-101
  apply common-inbound
  if community matches-any ([101..200]:201) then
    apply set-lpref-prepend(100,101,2)
    set community (2:1234) additive
  else
    set local-preference 125
  endif
end-policy
!
route-policy filter-bogons
  if destination in bogon then
drop
else
pass
  endif
end-policy
!
route-policy common-inbound
  apply filter-bogons
  set origin igp
  set community (2:333)
```

```
end-policy
!
route-policy set-lpref-prepend($lpref,$as,$prependcnt)
  set local-preference $lpref
  prepend as-path $as $prependcnt
end-policy
```

References for 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.

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, where element-entry is an entry of an item appropriate to the type of usage such as a prefix or a community value.

The following is an example using an inline community set:

```
route-policy sample-inline
if community matches-any ([10..15]:100) then
set local-preference 100
endif
end-policy
```

The following is an equivalent example using the named set test-communities:

```
community-set test-communities
10:100,
11:100,
12:100,
13:100,
14:100,
15:100
end-set

route-policy sample
if community matches-any test-communities then
set local-preference 100
endif
end-policy
```

Both of these policies are functionally equivalent, but the inline form does not require the configuration of the community set just to store the six values. You can choose the form appropriate to the configuration context. In the following sections, examples of both the named set version and the inline form are provided where appropriate.

as-path-set

An AS path set comprises operations for matching an AS path attribute. The only matching operation is a regular expression match.

Named Set Form

The named set form uses the **ios-regex** keyword to indicate the type of regular expression and requires single quotation marks around the regular expression.

The following is a sample definition of a named AS path set:

```
as-path-set aset1
ios-regex '_42$',
ios-regex '_127$'
end-set
```

This AS path set comprises two elements. When used in a matching operation, this AS path set matches any route whose AS path ends with either the autonomous system (AS) number 42 or 127.

To remove the named AS path set, use the **no as-path-set aset1** command-line interface (CLI) command.

Note Regular expression matching is CPU intensive. The policy performance can be substantially improved by either collapsing the regular expression patterns together to reduce the total number of regular expression invocations or by using equivalent native as-path match operations such as 'as-path neighbor-is', 'as-path originates-from' or 'as-path passes-through'.

Inline Set Form

The inline set form is a parenthesized list of comma-separated expressions, as follows:

```
(ios-regex '_42$', ios-regex '_127$')
```

This set matches the same AS paths as the previously named set, but does not require the extra effort of creating a named set separate from the policy that uses it.

community-set

A community-set holds community values for matching against the BGP community attribute. A community is a 32-bit quantity. Integer community values *must* be split in half and expressed as two unsigned decimal integers in the range from 0 to 65535, separated by a colon. Single 32-bit community values are not allowed. The following is the named set form:

Named Set Form

```
community-set cset1
12:34,
12:56,
12:78,
internet
end-set
```

Inline Set Form

```
(12:34, 12:56, 12:78)
($as:34, $as:$tag1, 12:78, internet)
```

The inline form of a community-set also supports parameterization. Each 16-bit portion of the community may be parameterized.

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 65535:65283.

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
```

Every community set must contain at least one community value. Empty community sets are invalid and are rejected.

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 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:

```
RP/0/RP0/CPU0:router(config)#extcommunity-set rt rt_set
RP/0/RP0/CPU0:router(config-ext)#?
  #-remark        Remark beginning with '#'
  *               Wildcard (any community or part thereof)
  <1-4294967295>  32-bit decimal number
  <1-65535>       16-bit decimal number
  A.B.C.D/M:N     Extended community - IPv4 prefix format
  A.B.C.D:N       Extended community - IPv4 format
  ASN:N           Extended community - ASPLAIN format
  X.Y:N           Extended community - ASDOT format
  abort           Discard RPL definition and return to top level config
  dfa-regex       DFA style regular expression
  end-set         End of set definition
  exit            Exit from this submode
  ios-regex       Traditional IOS style regular expression
  show            Show partial RPL configuration
```

Option	Description
#-remark	Remark beginning with '#'
*	Wildcard (any community or part thereof)
<1-4294967295>	32-bit decimal number
<1-65535>	16-bit decimal number
A.B.C.D/M:N	Extended community - IPv4 prefix format
A.B.C.D:N	Extended community - IPv4 format
ASN:N	Extended community - ASPLAIN format

Option	Description
X.Y:N	Extended community - ASDOT format
abort	Discard RPL definition and return to top level config
dfa-regex	DFA style regular expression
end-set	End of set definition
exit	Exit from this submode
ios-regex	Traditional IOS style regular expression
show	Show partial RPL configuration

Named Form for Extcommunity-set Soo

A soo set is an extcommunity set used to store BGP Site-of-Origin (SoO) extended community type communities:

```
  extcommunity-set soo a_soo_set
1.1.1:100,
     100:200
  end-set
```

These options are supported under extended community set Soo:

```
RP/0/RP0/CPU0:router(config)#extcommunity-set soo soo_set
RP/0/RP0/CPU0:router(config-ext)#?
  #-remark        Remark beginning with '#'
  *               Wildcard (any community or part thereof)
  <1-4294967295>  32-bit decimal number
  <1-65535>       16-bit decimal number
  A.B.C.D/M:N     Extended community - IPv4 prefix format
  A.B.C.D:N       Extended community - IPv4 format
  ASN:N           Extended community - ASPLAIN format
  X.Y:N           Extended community - ASDOT format
  abort           Discard RPL definition and return to top level config
  dfa-regex       DFA style regular expression
  end-set         End of set definition
  exit            Exit from this submode
  ios-regex       Traditional IOS style regular expression
  show            Show partial RPL configuration
```

Option	Description
#-remark	Remark beginning with '#'
*	Wildcard (any community or part thereof)
<1-4294967295>	32-bit decimal number
<1-65535>	16-bit decimal number
A.B.C.D/M:N	Extended community - IPv4 prefix format
A.B.C.D:N	Extended community - IPv4 format
ASN:N	Extended community - ASPLAIN format

Option	Description
X.Y:N	Extended community - ASDOT format
abort	Discard RPL definition and return to top level config
dfa-regex	DFA style regular expression
end-set	End of set definition
exit	Exit from this submode
ios-regex	Traditional IOS style regular expression
show	Show partial RPL configuration

prefix-set

A prefix-set holds IPv4 or IPv6 prefix match specifications, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. The address is a standard dotted-decimal IPv4 or colon-separated hexadecimal IPv6 address. The mask length, if present, is a nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6) following the address and separated from it by a slash. The optional minimum matching length follows the address and optional mask length and is expressed as the keyword **ge** (mnemonic for **g**reater than or **e**qual to), followed by a nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6). The optional maximum matching length follows the rest and is expressed by the keyword **le** (mnemonic for **l**ess than or **e**qual to), followed by yet another nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6). A syntactic shortcut for specifying an exact length for prefixes to match is the **eq** keyword (mnemonic for **eq**ual to).

If a prefix match specification has no mask length, then the default mask length is 32 for IPv4 and 128 for IPv6. The default minimum matching length is the mask length. If a minimum matching length is specified, then the default maximum matching length is 32 for IPv4 and 128 for IPv6. Otherwise, if neither minimum nor maximum is specified, the default maximum is the mask length.

The prefix-set itself is a comma-separated list of prefix match specifications. The following are examples:

```
prefix-set legal-ipv4-prefix-examples
   10.0.1.1,
   10.0.2.0/24,
   10.0.3.0/24 ge 28,
   10.0.4.0/24 le 28,
   10.0.5.0/24 ge 26 le 30,
   10.0.6.0/24 eq 28,
   10.0.7.2/32 ge 16 le 24,
   10.0.8.0/26 ge 8 le 16
end-set

prefix-set legal-ipv6-prefix-examples
  2001:0:0:1::/64,
  2001:0:0:2::/64 ge 96,
  2001:0:0:2::/64 ge 96 le 100,
  2001:0:0:2::/64 eq 100
end-set
```

The first element of the prefix-set matches only one possible value, 10.0.1.1/32 or the host address 10.0.1.1. The second element matches only one possible value, 10.0.2.0/24. The third element matches a range of prefix

values, from 10.0.3.0/28 to 10.0.3.255/32. The fourth element matches a range of values, from 10.0.4.0/24 to 10.0.4.240/28. The fifth element matches prefixes in the range from 10.0.5.0/26 to 10.0.5.252/30. The sixth element matches any prefix of length 28 in the range from 10.0.6.0/28 through 10.0.6.240/28. The seventh element matches any prefix of length 32 in the range 10.0.[0..255].2/32 (from 10.0.0.2/32 to 10.0.255.2). The eighth element matches any prefix of length 26 in the range 10.[0..255].8.0/26 (from 10.0.8.0/26 to 10.255.8.0/26).

The following prefix-set consists entirely of invalid prefix match specifications:

```
prefix-set ILLEGAL-PREFIX-EXAMPLES
   10.1.1.1 ge 16,
   10.1.2.1 le 16,
   10.1.3.0/24 le 23,
   10.1.4.0/24 ge 33,
   10.1.5.0/25 ge 29 le 28
end-set
```

Neither the minimum length nor maximum length is valid without a mask length. For IPv4, the minimum length must be less than 32, the maximum length of an IPv4 prefix. For IPv6, the minimum length must be less than 128, the maximum length of an IPv6 prefix. The maximum length must be equal to or greater than the minimum length.

ACL Support in RPL Prefix Sets

Access Control List (ACL) type prefix set entries holds IPv4 or IPv6 prefix match specifications, each of which has an address and a wildcard mask. The address and wildcard mask is a standard dotted-decimal IPv4 or colon-separated hexadecimal IPv6 address. The set of bits to be matched are provided in the form of wildcard also called as inverted mask in which a binary 0 means a mandatory match and binary 1 means a do not match condition. The prefix set allows to specify contiguous and non-contiguous set of bits that should be matched in any route.

rd-set

An rd-set is used to create a set with route distinguisher (RD) elements. An RD set is a 64-bit value prepended to an IPv4 address to create a globally unique Border Gateway Protocol (BGP) VPN IPv4 address.

You can define RD values with the following commands:

- *a.b.c.d:m:**—BGP VPN RD in IPv4 format with a wildcard character. For example, 10.0.0.2:255.255.0.0:*.
- *a.b.c.d/m:n*—BGP VPN RD in IPv4 format with a mask. For example, 10.0.0.2:255.255.0.0:666.
- *a.b.c.d:*** —BGP VPN RD in IPv4 format with a wildcard character. For example, 10.0.0.2:255.255.0.0.
- *a.b.c.d:n*— BGP VPN RD in IPv4 format. For example, 10.0.0.2:666.
- *asn:**— BGP VPN RD in ASN format with a wildcard character. For example, 10002:255.255.0.0.
- *asn:n*—BGP VPN RD in ASN format. For example, 10002:666.

The following is an example of an rd-set:

```
rd-set rdset1
   10.0.0.0/8:*,
   10.0.0.0/8:777,
   10.0.0.0:*,
```

```
    10.0.0.0:777,
    65000:*,
    65000:777
end-set
```

Routing Policy Language Components

Four main components in the routing policy language are involved in defining, modifying, and using policies: the configuration front end, policy repository, execution engine, and policy clients themselves.

The configuration front end (CLI) is the mechanism to define and modify policies. This configuration is then stored on the router using the normal storage means and can be displayed using the normal configuration **show** commands.

The second component of the policy infrastructure, the policy repository, has several responsibilities. First, it compiles the user-entered configuration into a form that the execution engine can understand. Second, it performs much of the verification of policies; and it ensures that defined policies can actually be executed properly. Third, it tracks which attach points are using which policies so that when policies are modified the appropriate clients are properly updated with the new policies relevant to them.

The third component is the execution engine. This component is the piece that actually runs policies as the clients request. The process can be thought of as receiving a route from one of the policy clients and then executing the actual policy against the specific route data.

The fourth component is the policy clients (the routing protocols). This component calls the execution engine at the appropriate times to have a given policy be applied to a given route, and then perform some number of actions. These actions may include deleting the route if policy indicated that it should be dropped, passing along the route to the protocol decision tree as a candidate for the best route, or advertising a policy modified route to a neighbor or peer as appropriate.

Routing Policy Language Usage

This section provides basic routing policy language usage examples.

Pass PolicyPass Policy

The following example shows how the policy accepts all presented routes without modifying the routes.

```
route-policy quickstart-pass
pass
end-policy
```

Drop Everything Policy

The following example shows how the policy explicitly rejects all routes presented to it. This type of policy is used to ignore everything coming from a specific peer.

```
route-policy quickstart-drop
drop
end-policy
```

Ignore Routes with Specific AS Numbers in the Path

The following example shows the policy definition in three parts. First, the **as-path-set** command defines three regular expressions to match against an AS path. Second, the **route-policy** command applies the AS path set to a route. If the AS path attribute of the route matches the regular expression defined with the **as-path-set** command, the protocol refuses the route. Third, the route policy is attached to BGP neighbor 10.0.1.2. BGP consults the policy named ignore_path_as on routes received (imported) from neighbor 10.0.1.2.

```
as-path-set ignore_path
ios-regex '_11_',
ios-regex '_22_',
ios-regex '_33_'
end-set

route-policy ignore_path_as
if as-path in ignore_path then
drop
else
pass
endif
end-policy

router bgp 2
neighbor 10.0.1.2 address-family ipv4 unicast policy ignore_path_as in
```

Set Community Based on MED

The following example shows how the policy tests the MED of a route and modifies the community attribute of the route based on the value of the MED. If the MED value is 127, the policy adds the community 123:456 to the route. If the MED value is 63, the policy adds the value 123:789 to the community attribute of the route. Otherwise, the policy removes the community 123:123 from the route. In any case, the policy instructs the protocol to accept the route.

```
route-policy quickstart-med
if med eq 127 then
set community (123:456) additive
elseif med eq 63 then
set community (123:789) additive
else
delete community in (123:123)
endif
pass
end-policy
```

Set Local Preference Based on Community

The following example shows how the community-set named quickstart-communities defines community values. The route policy named quickstart-localpref tests a route for the presence of the communities specified in the quickstart-communities community set. If any of the community values are present in the route, the route policy sets the local preference attribute of the route to 31. In any case, the policy instructs the protocol to accept the route.

```
community-set quickstart-communities
987:654,
987:543,
987:321,
```

```
987:210
end-set

route-policy quickstart-localpref
if community matches-any quickstart-communities then
set local-preference 31
endif
pass
end-policy
```

Persistent Remarks

The following example shows how comments are placed in the policy to clarify the meaning of the entries in the set and the statements in the policy. The remarks are persistent, meaning they remain attached to the policy. For example, remarks are displayed in the output of the **show running-config** command. Adding remarks to the policy makes the policy easier to understand, modify at a later date, and troubleshoot if an unexpected behavior occurs.

```
prefix-set rfc1918
# These are the networks defined as private in RFC1918 (including
# all subnets thereof)
10.0.0.0/8 ge 8,
172.16.0.0/12 ge 12,
192.168.0.0/16 ge 16
end-set

route-policy quickstart-remarks
# Handle routes to RFC1918 networks
if destination in rfc1918 then
# Set the community such that we do not export the route
set community (no-export) additive

endif
end-policy
```

Policy Definitions

Policy definitions create named sequences of policy statements. A policy definition consists of the CLI **route-policy** keyword followed by a name, a sequence of policy statements, and the **end-policy** keyword. For example, the following policy drops any route it encounters:

```
route-policy drop-everything
drop
end-policy
```

The name serves as a handle for binding the policy to protocols. To remove a policy definition, issue the **no route-policy** *name* command.

Policies may also refer to other policies such that common blocks of policy can be reused. This reference to other policies is accomplished by using the **apply** statement, as shown in the following example:

```
route-policy check-as-1234
if as-path passes-through '1234.5' then
apply drop-everything
```

```
else
pass
endif
end-policy
```

The **apply** statement indicates that the policy drop-everything should be executed if the route under consideration passed through autonomous system 1234.5 before it is received. If a route that has autonomous system 1234.5 in its AS path is received, the route is dropped; otherwise, the route is accepted without modification. This policy is an example of a hierarchical policy. Thus, the semantics of the **apply** statement are just as if the applied policy were cut and pasted into the applying policy:

```
route-policy check-as-1234-prime
 if as-path passes-through '1234.5' then
    drop
 else
    pass
 endif
end-policy
```

You may have as many levels of hierarchy as desired. However, many levels may be difficult to maintain and understand.

Parameterization

In addition to supporting reuse of policies using the **apply** statement, policies can be defined that allow for parameterization of some of the attributes. The following example shows how to define a parameterized policy named param-example. In this case, the policy takes one parameter, $mytag. Parameters always begin with a dollar sign and consist otherwise of any alphanumeric characters. Parameters can be substituted into any attribute that takes a parameter.

In the following example, a 16-bit community tag is used as a parameter:

```
route-policy param-example ($mytag)
set community (1234:$mytag) additive
end-policy
```

This parameterized policy can then be reused with different parameterization, as shown in the following example. In this manner, policies that share a common structure but use different values in some of their individual statements can be modularized. For details on which attributes can be parameterized, see the individual attribute sections.

```
route-policy origin-10
if as-path originates-from ‘10.5’ then
apply param-example(10.5)
else
pass
endif
end-policy

route-policy origin-20
if as-path originates-from ‘20.5’ then
apply param-example(20.5)
```

```
else
pass
endif
end-policy
```

The parameterized policy param-example provides a policy definition that is expanded with the values provided as the parameters in the apply statement. Note that the policy hierarchy is always maintained, Thus, if the definition of param-example changes, then the behavior of origin_10 and origin_20 changes to match.

The effect of the origin-10 policy is that it adds the community 1234:10 to all routes that pass through this policy and have an AS path indicating the route originated from autonomous system 10. The origin-20 policy is similar except that it adds to community 1234:20 for routes originating from autonomous system 20.

Parameterization at Attach Points

In addition to supporting parameterization using the apply statement, policies can also be defined that allow for parameterization the attributes at attach points. Parameterization is supported at all attach points.

In the following example, we define a parameterized policy "param-example". In this example, the policy takes two parameters "$mymed" and “$prefixset”. Parameters always begin with a dollar sign, and consist otherwise of any alphanumeric characters. Parameters can be substituted into any attribute that takes a parameter. In this example we are passing a MED value and prefix set name as parameters.

```
route-policy param-example ($mymed, $prefixset)
  if destination in $prefixset then
  set med $mymed
  endif
end-policy
```

This parameterized policy can then be reused with different parameterizations as shown in the example below. In this manner, policies that share a common structure but use different values in some of their individual statements can be modularized. For details on which attributes can be parameterized, see the individual attributes for each protocol.

```
router bgp 2
  neighbor 10.1.1.1
    remote-as 3
    address-family ipv4 unicast
      route-policy param-example(10, prefix_set1)
      route-policy param-example(20, prefix_set2)
```

The parameterized policy param-example provides a policy definition that is expanded with the values provided as the parameters in the neighbor route-policy in and out statement.

Global Parameterization

RPL supports the definition of systemwide global parameters that can be used inside policy definition. Global parameters can be configured as follows:

```
Policy-global
  glbpathtype ‘ebgp’
  glbtag ‘100’
```

```
end-global
```

The global parameter values can be used directly inside a policy definition similar to the local parameters of parameterized policy. In the following example, the *globalparam* argument, which makes use of the global parameters gbpathtype and glbtag, is defined for a nonparameterized policy.

```
route-policy globalparam
  if path-type is $glbpathtype then
    set tag $glbtag
  endif
end-policy
```

When a parameterized policy has a parameter name “collision” with a global parameter name, parameters local to policy definition take precedence, effectively masking off global parameters. In addition, a validation mechanism is in place to prevent the deletion of a particular global parameter if it is referred by any policy.

Semantics of Policy Application

This section discusses how routing policies are evaluated and applied. The following concepts are discussed:

Boolean Operator Precedence

Boolean expressions are evaluated in order of operator precedence, from left to right. The highest precedence operator is NOT, followed by AND, and then OR. The following expression:

```
med eq 10 and not destination in (10.1.3.0/24) or community matches-any ([10..25]:35)
```

if fully parenthesized to display the order of evaluation, would look like this:

```
(med eq 10 and (not destination in (10.1.3.0/24))) or community matches-any ([10..25]:35)
```

The inner NOT applies only to the destination test; the AND combines the result of the NOT expression with the Multi Exit Discriminator (MED) test; and the OR combines that result with the community test. If the order of operations are rearranged:

```
not med eq 10 and destination in (10.1.3.0/24) or community matches-any ([10..25]:35)
```

then the expression, fully parenthesized, would look like the following:

```
((not med eq 10) and destination in (10.1.3.0/24)) or community matches-any ([10..25]:35)
```

Multiple Modifications of Same Attribute

When a policy replaces the value of an attribute multiple times, the last assignment wins because all actions are executed. Because the MED attribute in BGP is one unique value, the last value to which it gets set to wins. Therefore, the following policy results in a route with a MED value of 12:

```
set med 9
set med 10
set med 11
set med 12
```

This example is trivial, but the feature is not. It is possible to write a policy that effectively changes the value for an attribute. For example:

```
set med 8
if community matches-any cs1 then
set local-preference 122
if community matches-any cs2 then
set med 12
endif
endif
```

The result is a route with a MED of 8, unless the community list of the route matches both cs1 and cs2, in which case the result is a route with a MED of 12.

In the case in which the attribute being modified can contain only one value, it is easy to think of this case as the last statement wins. However, a few attributes can contain multiple values and the result of multiple actions on the attribute is cumulative rather than as a replacement. The first of these cases is the use of the **additive** keyword on community and extended community evaluation. Consider a policy of the form:

```
route-policy community-add
set community (10:23)
set community (10:24) additive
set community (10:25) additive
end-policy
```

This policy sets the community string on the route to contain all three community values: 10:23, 10:24, and 10:25.

The second of these cases is AS path prepending. Consider a policy of the form:

```
route-policy prepend-example
prepend as-path 2.5 3
prepend as-path 666.5 2
end-policy
```

This policy prepends 666.5 666.5 2.5 2.5 2.5 to the AS path. This prepending is a result of all actions being taken and to the AS path being an attribute that contains an array of values rather than a simple scalar value.

When Attributes Are Modified

A policy does not modify route attribute values until all tests have been completed. In other words, comparison operators always run on the initial data in the route. Intermediate modifications of the route attributes do not have a cascading effect on the evaluation of the policy. Take the following example:

```
ifmed eq 12 then
```

```
set med 42
if med eq 42 then
drop
endif
endif
```

This policy never executes the drop statement because the second test (med eq 42) sees the original, unmodified value of the MED in the route. Because the MED has to be 12 to get to the second test, the second test always returns false.

Default Drop Disposition

All route policies have a default action to drop the route under evaluation unless the route has been modified by a policy action or explicitly passed. Applied (nested) policies implement this disposition as though the applied policy were pasted into the point where it is applied.

Consider a policy to allow all routes in the 10 network and set their local preference to 200 while dropping all other routes. You might write the policy as follows:

```
route-policy two
if destination in (10.0.0.0/8 ge 8 le 32) then
set local-preference 200
endif
end-policy

route-policy one
apply two
end-policy
```

It may appear that policy one drops all routes because it neither contains an explicit **pass** statement nor modifies a route attribute. However, the applied policy does set an attribute for some routes and this disposition is passed along to policy one. The result is that policy one passes routes with destinations in network 10, and drops all others.

Control Flow

Policy statements are processed sequentially in the order in which they appear in the configuration. Policies that hierarchically reference other policy blocks are processed as if the referenced policy blocks had been directly substituted inline. For example, if the following policies are defined:

```
route-policy one
set weight 100
end-policy

route-policy two
set med 200
end-policy

route-policy three
apply two
set community (2:666) additive
end-policy

route-policy four
apply one
```

```
apply three
pass
end-policy
```

Policy four could be rewritten in an equivalent way as follows:

```
route-policy four-equivalent
set weight 100
set med 200
set community (2:666) additive
pass
end-policy
```

Note The **pass** statement is not required and can be removed to represent the equivalent policy in another way.

Policy Verification

Several different types of verification occur when policies are being defined and used.

Range Checking

As policies are being defined, some simple verifications, such as range checking of values, is done. For example, the MED that is being set is checked to verify that it is in a proper range for the MED attribute. However, this range checking cannot cover parameter specifications because they may not have defined values yet. These parameter specifications are verified when a policy is attached to an attach point. The policy repository also verifies that there are no recursive definitions of policy, and that parameter numbers are correct. At attach time, all policies must be well formed. All sets and policies that they reference must be defined and have valid values. Likewise, any parameter values must also be in the proper ranges.

Incomplete Policy and Set References

As long as a given policy is not attached at an attach point, the policy is allowed to refer to nonexistent sets and policies, which allows for freedom of workflow. You can build configurations that reference sets or policy blocks that are not yet defined, and then can later fill in those undefined policies and sets, thereby achieving much greater flexibility in policy definition. Every piece of policy you want to reference while defining a policy need not exist in the configuration. Thus, a user can define a policy sample that references the policy bar using an **apply** statement even if the policy bar does not exist. Similarly, a user can enter a policy statement that refers to a nonexistent set.

However, the existence of all referenced policies and sets is enforced when a policy is attached. If you attempt to attach the policy sample with the reference to an undefined policy bar at an inbound BGP policy using the **neighbor 1.2.3.4 address-family ipv4 unicast policy sample in** command, the configuration attempt is rejected because the policy bar does not exist.

Likewise, you cannot remove a route policy or set that is currently in use at an attach point because this removal would result in an undefined reference. An attempt to remove a route policy or set that is currently in use results in an error message to the user.

A condition exists that is referred to as a null policy in which the policy bar exists but has no statements, actions, or dispositions in it. In other words, the policy bar does exist as follows:

```
route-policy bar
end-policy
```

This is a valid policy block. It effectively forces all routes to be dropped because it is a policy block that never modifies a route, nor does it include the pass statement. Thus, the default action of drop for the policy block is followed.

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.

Policy Statements

Four types of policy statements exist: remark, disposition (drop and pass), action (set), and if (comparator).

Remark

A remark is text attached to policy configuration but otherwise ignored by the policy language parser. Remarks are useful for documenting parts of a policy. The syntax for a remark is text that has each line prepended with a pound sign (#):

```
# This is a simple one-line remark.

# This
# is a remark
# comprising multiple
# lines.
```

In general, remarks are used between complete statements or elements of a set. Remarks are not supported in the middle of statements or within an inline set definition.

Unlike traditional !-comments in the CLI, RPL remarks persist through reboots and when configurations are saved to disk or a TFTP server and then loaded back onto the router.

Disposition

If a policy modifies a route, by default the policy accepts the route. RPL provides a statement to force the opposite—the **drop** statement. If a policy matches a route and executes a drop, the policy does not accept the route. If a policy does not modify the route, by default the route is dropped. To prevent the route from being dropped, the **pass** statement is used.

The **drop** statement indicates that the action to take is to discard the route. When a route is dropped, no further execution of policy occurs. For example, if after executing the first two statements of a policy the **drop** statement is encountered, the policy stops and the route is discarded.

Note All policies have a default **drop** action at the end of execution.

The **pass** statement allows a policy to continue executing even though the route has not been modified. When a policy has finished executing, any route that has been modified in the policy or any route that has received a pass disposition in the policy, successfully passes the policy and completes the execution. If route policy

B_rp is applied within route policy A_rp, execution continues from policy A_rp to policy B_rp and back to policy A_rp provided prefix is not dropped by policy B_rp.

```
route-policy A_rp
  set community (10:10)
  apply B_rp
end-policy
!

route-policy B_rp
  if destination in (121.23.0.0/16 le 32, 155.12.0.0/16 le 32) then
  set community (121:155) additive
  endif
end-policy
!
```

By default, a route is **dropped** at the end of policy processing unless either the policy **modifies** a route attribute or it passes the route by means of an explicit **pass** statement. For example, if route-policy B is applied within route-policy A, then execution continues from policy A to policy B and back to policy A, provided the prefix is not dropped by policy B.

```
route-policy A
   if as-path neighbor-is '123' then
   apply B
   policy statement N
end-policy
```

Whereas the following policies pass all routes that they evaluate.

```
route-policy PASS-ALL
pass
end-policy
```

```
route-policy SET-LPREF
set local-preference 200
end-policy
```

In addition to being implicitly dropped, a route may be dropped by an **explicit drop** statement. **Drop** statements cause a route to be dropped immediately so that no further policy processing is done. Note also that a **drop** statement overrides any previously processed **pass** statements or attribute modifications. For example, the following policy drops all routes. The first **pass** statement is executed, but is then immediately overridden by the **drop** statement. The second **pass** statement never gets executed.

```
route-policy DROP-EXAMPLE
pass
drop
pass
end-policy
```

When one policy applies another, it is as if the applied policy were copied into the right place in the applying policy, and then the same drop-and-pass semantics are put into effect. For example, policies ONE and TWO are equivalent to policy ONE-PRIME:

```
route-policy ONE
apply two
if as-path neighbor-is '123' then
pass
endif
end-policy

route-policy TWO
if destination in (10.0.0.0/16 le 32) then
drop
endif
end-policy

route-policy ONE-PRIME
if destination in (10.0.0.0/16 le 32) then
drop
endif
if as-path neighbor-is '123' then
pass
endif
end-policy
```

Because the effect of an **explicit drop** statement is immediate, routes in 10.0.0.0/16 le 32 are dropped without any further policy processing. Other routes are then considered to see if they were advertised by autonomous system 123. If they were advertised, they are passed; otherwise, they are implicitly dropped at the end of all policy processing.

The **done** statement indicates that the action to take is to stop executing the policy and accept the route. When encountering a **done** statement, the route is passed and no further policy statements are executed. All modifications made to the route prior to the **done** statement are still valid.

Action

An action is a sequence of primitive operations that modify a route. Most actions, but not all, are distinguished by the **set** keyword. In a route policy, actions can be grouped together. For example, the following is a route policy comprising three actions:

```
route-policy actions
set med 217
set community (12:34) additive
delete community in (12:56)
end-policy
```

If

In its simplest form, an **if** statement uses a conditional expression to decide which actions or dispositions should be taken for the given route. For example:

```
if as-path in as-path-set-1 then
drop
endif
```

The example indicates that any routes whose AS path is in the set as-path-set-1 are dropped. The contents of the **then** clause may be an arbitrary sequence of policy statements.

The following example contains two action statements:

```
if origin is igp then
set med 42
prepend as-path 73.5 5
endif
```

The CLI provides support for the **exit** command as an alternative to the **endif** command.

The **if** statement also permits an **else** clause, which is executed if the if condition is false:

```
if med eq 8 then
set community (12:34) additive
else
set community (12:56) additive
endif
```

The policy language also provides syntax, using the **elseif** keyword, to string together a sequence of tests:

```
if med eq 150 then
set local-preference 10
elseif med eq 200 then
set local-preference 60
elseif med eq 250 then
set local-preference 110
else
set local-preference 0
endif
```

The statements within an **if** statement may themselves be **if** statements, as shown in the following example:

```
if community matches-any (12:34,56:78) then
if med eq 150 then
drop
endif
set local-preference 100
endif
```

This policy example sets the value of the local preference attribute to 100 on any route that has a community value of 12:34 or 56:78 associated with it. However, if any of these routes has a MED value of 150, then these routes with either the community value of 12:34 or 56:78 and a MED of 150 are dropped.

Boolean Conditions

In the previous section describing the **if** statement, all of the examples use simple Boolean conditions that evaluate to either true or false. RPL also provides a way to build compound conditions from simple conditions by means of Boolean operators.

Three Boolean operators exist: negation (**not**), conjunction (**and**), and disjunction (**or**). In the policy language, negation has the highest precedence, followed by conjunction, and then by disjunction. Parentheses may be used to group compound conditions to override precedence or to improve readability.

The following simple condition:

```
med eq 42
```

is true only if the value of the MED in the route is 42, otherwise it is false.

A simple condition may also be negated using the **not** operator:

```
not next-hop in (10.0.2.2)
```

Any Boolean condition enclosed in parentheses is itself a Boolean condition:

```
(destination in prefix-list-1)
```

A compound condition takes either of two forms. It can be a simple expression followed by the **and** operator, itself followed by a simple condition:

```
med eq 42 and next-hop in (10.0.2.2)
```

A compound condition may also be a simpler expression followed by the **or** operator and then another simple condition:

```
origin is igp or origin is incomplete
```

An entire compound condition may be enclosed in parentheses:

```
(med eq 42 and next-hop in (10.0.2.2))
```

The parentheses may serve to make the grouping of subconditions more readable, or they may force the evaluation of a subcondition as a unit.

In the following example, the highest-precedence **not** operator applies only to the destination test, the **and** operator combines the result of the **not** expression with the community test, and the **or** operator combines that result with the MED test.

```
med eq 10 or not destination in (10.1.3.0/24) and community matches-any ([12..34]:[56..78])
```

With a set of parentheses to express the precedence, the result is the following:

```
med eq 10 or ((not destination in (10.1.3.0/24)) and community matches-any
```

```
([12..34]:[56..78])
```

The following is another example of a complex expression:

```
(origin is igp or origin is incomplete or not med eq 42) and next-hop in (10.0.2.2)
```

The left conjunction is a compound condition enclosed in parentheses. The first simple condition of the inner compound condition tests the value of the origin attribute; if it is Interior Gateway Protocol (IGP), then the inner compound condition is true. Otherwise, the evaluation moves on to test the value of the origin attribute again, and if it is incomplete, then the inner compound condition is true. Otherwise, the evaluation moves to check the next component condition, which is a negation of a simple condition.

apply

As discussed in the sections on policy definitions and parameterization of policies, the **apply** command executes another policy (either parameterized or unparameterized) from within another policy, which allows for the reuse of common blocks of policy. When combined with the ability to parameterize common blocks of policy, the **apply** command becomes a powerful tool for reducing repetitive configuration.

Attach Points

Policies do not become useful until they are applied to routes, and for policies to be applied to routes they need to be made known to routing protocols. In BGP, for example, there are several situations where policies can be used, the most common of these is defining import and export policy. The policy attach point is the point in which an association is formed between a specific protocol entity, in this case a BGP neighbor, and a specific named policy. It is important to note that a verification step happens at this point. Each time a policy is attached, the given policy and any policies it may apply are checked to ensure that the policy can be validly used at that attach point. For example, if a user defines a policy that sets the IS-IS level attribute and then attempts to attach this policy as an inbound BGP policy, the attempt would be rejected because BGP routes do not carry IS-IS attributes. Likewise, when policies are modified that are in use, the attempt to modify the policy is verified against all current uses of the policy to ensure that the modification is compatible with the current uses.

Each protocol has a distinct definition of the set of attributes (commands) that compose a route. For example, BGP routes may have a community attribute, which is undefined in OSPF. Routes in IS-IS have a level attribute, which is unknown to BGP. Routes carried internally in the RIB may have a tag attribute.

When a policy is attached to a protocol, the protocol checks the policy to ensure the policy operates using route attributes known to the protocol. If the protocol uses unknown attributes, then the protocol rejects the attachment. For example, OSPF rejects attachment of a policy that tests the values of BGP communities.

The situation is made more complex by the fact that each protocol has access to at least two distinct route types. In addition to native protocol routes, for example BGP or IS-IS, some protocol policy attach points operate on RIB routes, which is the common central representation. Using BGP as an example, the protocol provides an attach point to apply policy to routes redistributed from the RIB to BGP. An attach point dealing with two different kinds of routes permits a mix of operations: RIB attribute operations for matching and BGP attribute operations for setting.

Note The protocol configuration rejects attempts to attach policies that perform unsupported operations.

The following sections describe the protocol attach points, including information on the attributes (commands) and operations that are valid for each attach point.

BGP Policy Attach Points

This section describes each of the BGP policy attach points and provides a summary of the BGP attributes and operators.

Additional-Path

The additional-path attach point provides increased control based on various attribute match operations. This attach point is used to decide whether a route-policy should be used to select additional-paths for a BGP speaker to be able to send multiple paths for the prefix.

The add path enables BGP prefix independent convergence (PIC) at the edge routers.

This example shows how to set a route-policy "add-path-policy" to be used for enabling selection of additional paths:

```
router bgp 100
 address-family ipv4 unicast
 additional-paths selection route-policy add-path-policy
```

Dampening

The dampening attach point controls the default route-dampening behavior within BGP. Unless overridden by a more specific policy on the associate peer, all routes in BGP apply the associated policy to set their dampening attributes.

The following policy sets dampening values for BGP IPv4 unicast routes. Those routes that are more specific than a /25 take longer to recover after they are dampened than the routes that are less specific than /25.

Note When the dampening policy runs for a route, then the last "set dampening" statement that is encountered, takes effect.

- If a "drop" statement is encountered, then the route is not dampened; even if the "set dampening" statement is encountered.
- If a "pass" or "done" statement is encountered but not the "set dampening" statement, then the route is dampened using the default dampening parameters.

For example:

- When policy1 applies another policy that is called policy2 and if a "pass" statement is encountered in policy2, then policy2 exits and continues to execute policy1.
- If a "done" statement is encountered in policy2, then both policy1 and policy2 exits immediately.

```
route-policy sample_damp
  if destination in (0.0.0.0/0 ge 25) then
    set dampening halflife 30 others default
  else
    set dampening halflife 20 others default
  endif
end-policy

router bgp 2
  address-family ipv4 unicast
    bgp dampening route-policy sample_damp
    .
    .
    .
```

Default Originate

The default originate attach point allows the default route (0.0.0.0/0) to be conditionally generated and advertised to a peer, based on the presence of other routes. It accomplishes this configuration by evaluating the associated policy against routes in the Routing Information Base (RIB). If any routes pass the policy, the default route is generated and sent to the relevant peer.

The following policy generates and sends a default-route to the BGP neighbor 10.0.0.1 if any routes that match 10.0.0.0/8 ge 8 le 32 are present in the RIB.

```
route-policy sample-originate
  if rib-has-route in (10.0.0.0/8 ge 8 le 32) then
     pass
  endif
end-policy

router bgp 2
  neighbor 10.0.0.1
    remote-as 3
    address-family ipv4 unicast
    default-originate policy sample-originate
    .
    .
    .
```

Neighbor Export

The neighbor export attach point selects the BGP routes to send to a given peer or group of peers. The routes are selected by running the set of possible BGP routes through the associated policy. Any routes that pass the policy are then sent as updates to the peer or group of peers. The routes that are sent may have had their BGP attributes altered by the policy that has been applied.

The following policy sends all BGP routes to neighbor 10.0.0.5. Routes that are tagged with any community in the range 2:100 to 2:200 are sent with a MED of 100 and a community of 2:666. The rest of the routes are sent with a MED of 200 and a community of 2:200.

```
route-policy sample-export
  if community matches-any (2:[100-200]) then
    set med 100
    set community (2:666)
  else
    set med 200
```

```
    set community (2:200)
  endif
end-policy

router bgp 2
  neighbor 10.0.0.5
    remote-as 3
  address-family ipv4 unicast
    route-policy sample-export out
    .
    .
    .
```

Neighbor Import

The neighbor import attach point controls the reception of routes from a specific peer. All routes that are received by a peer are run through the attached policy. Any routes that pass the attached policy are passed to the BGP Routing Information Base (BRIB) as possible candidates for selection as best path routes.

When a BGP import policy is modified, it is necessary to rerun all the routes that have been received from that peer against the new policy. The modified policy may now discard routes that were previously allowed through, allow through previously discarded routes, or change the way the routes are modified. A new configuration option in BGP (**bgp auto-policy-soft-reset**) that allows this modification to happen automatically in cases for which either soft reconfiguration is configured or the BGP route-refresh capability has been negotiated.

The following example shows how to receive routes from neighbor 10.0.0.1. Any routes received with the community 3:100 have their local preference set to 100 and their community tag set to 2:666. All other routes received from this peer have their local preference set to 200 and their community tag set to 2:200.

```
route-policy sample_import
  if community matches-any (3:100) then
    set local-preference 100
    set community (2:666)
  else
    set local-preference 200
    set community (2:200)
  endif
end-policy

router bgp 2
  neighbor 10.0.0.1
    remote-as 3
    address-family ipv4 unicast
      route-policy sample_import in
      .
      .
      .
```

Network

The network attach point controls the injection of routes from the RIB into BGP. A route policy attached at this point is able to set any of the valid BGP attributes on the routes that are being injected.

The following example shows a route policy attached at the network attach point that sets the well-known community no-export for any routes more specific than /24:

```
route-policy NetworkControl
```

```
  if destination in (0.0.0.0/0 ge 25) then
    set community (no-export) additive
  endif
end-policy

router bgp 2
  address-family ipv4 unicast
    network 172.16.0.5/27 route-policy NetworkControl
```

Redistribute

The redistribute attach point 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
    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.

When you configure import route policy for a particular VRF, you must define the import route-target values. Configuring **import route-policy** command does not take effect until you configure the **import route-target** command with the route-target value. The import route target value acts as a first-level filter. The import policy that you configure using the **import route-policy** command acts as a second-level filter.

```
route-policy bgpvrf_import
  if extcommunity rt matches-any (10:91) then
    set next-hop 172.16.0.1
  elseif extcommunity rt matches-every (11:92) then
    set next-hop 172.16.0.2
  elseif extcommunity soo matches-any (10:111111, 10:111222) then
    pass
  endif
end-policy

vrf vrf_import
  address-family ipv4 unicast
```

```
      import route-policy bgpvrf_import
      .
      .
      .

import route-target
         65001:2200
       !
          export route-target
          65001:2201
```

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
    .
    .
    .
```

Retain Route-Target

The retain route target attach point within BGP allows the specification of match criteria based only on route target extended community. The attach point is useful at the route reflector (RR) or at the Autonomous System Boundary Router (ASBR).

Typically, an RR has to retain all IPv4 VPN routes to peer with its PE routers. These PEs might require routers tagged with different route target IPv4 VPN routes resulting in non-scalable RRs. You can achieve scalability if you configure an RR to retain routes with a defined set of route target extended communities, and a specific set of VPNs to service.

Another reason to use this attach point is for an ASBR. ASBRs do not require that VRFs be configured, but need this configuration to retain the IPv4 VPN prefix information.

The following example shows how to configure the route policy retainer and apply it to the retain route target attach point. The route is accepted if the route contains route target extended communities 10:615, 10:6150, and 15.15.15.15.:15. All other non-matching routes are dropped.

```
extcommunity-set rt rtset1
  0:615,
  10:6150,

..15.15.15.15.:15
  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.

Label-Mode

The label-mode attachpoint provides facility to choose label mode based on arbitrary match criteria such as prefix value, community. This attach point is typically used to set the type of label mode to per-ce or per-vrf or per-prefix based on deployment preferences. The attribute setting actions supported are for pass and drop.

Neighbor-ORF

The neighbor-orf attach point provides the filtering of incoming BGP route updates using only prefix-based matching. In addition to using this as an inbound filter, the prefixes and disposition (drop or pass) are sent to upstream neighbors as an Outbound Route Filter (ORF) to allow them to perform filtering.

The following example shows how to configure a route policy orf-preset and apply it to the neighbor ORF attach point. The prefix of the route is dropped if it matches any prefix specified in orf-preset (172.16.1.0/24, 172.16.5.0/24, 172.16.11.0/24). In addition to this inbound filtering, BGP also sends these prefix entries to the upstream neighbor with a permit or deny so that the neighbor can filter updates before sending them on to their destination.

```
prefix-set orf-preset
  172.16.1.0/24,
  172.16.5.0/24,
  172.16.11.0/24
end-set
```

```
route-policy policy-orf
  if orf prefix in orf-preset then
    drop
  endif
  if orf prefix in (172.16.3.0/24, 172.16.7.0/24, 172.16.13.0/24) then
    pass
  endif

router bgp 2
  neighbor 1.1.1.1
    remote-as 3
    address-family ipv4 unicast
      orf route-policy policy-orf
    .
    .
    .
```

Next-hop

The next-hop attach point provides increased control based on protocol and prefix-based match operations. The attach point is typically used to decide whether to act on a next-hop notification (up or down) event.

Support for next-hop tracking allows BGP to monitor reachability for routes in the Routing Information Base (RIB) that can directly affect BGP prefixes. The route policy at the BGP next-hop attach point helps limit notifications delivered to BGP for specific prefixes. The route policy is applied on RIB routes. Typically, route policies are used in conjunction with next-hop tracking to monitor non-BGP routes.

The following example shows how to configure the BGP next-hop tracking feature using a route policy to monitor static or connected routes with the prefix 10.0.0.0 and prefix length 8.

```
route-policy nxthp_policy_A
  if destination in (10.0.0.0/8) and protocol in (static, connected) then
    pass
  endif
end-policy

router bgp 2
  address-family ipv4 unicast
    nexthop route-policy nxthp_policy_A
      .
      .
      .
```

Clear-Policy

The clear-policy attach point provides increased control based on various AS path match operations when using a **clear bgp** command. This attach point is typically used to decide whether to clear BGP flap statistics based on AS-path-based match operations.

The following example shows how to configure a route policy where the in operator evaluates to true if one or more of the regular expression matches in the set my-as-set successfully match the AS path associated with the route. If it is a match, then the **clear** command clears the associated flap statistics.

```
as-path-set my-as-set
  ios-regex '_12$',
  ios-regex '_13$'
end-set
```

```
route-policy policy_a
  if as-path in my-as-set then
    pass
  else
    drop
  endif
end-policy

clear bgp ipv4 unicast flap-statistics route-policy policy_a
```

Debug

The debug attach point provides increased control based on prefix-based match operations. This attach point is typically used to filter debug output for various BGP commands based on the prefix of the route.

The following example shows how to configure a route policy that will only pass the prefix 20.0.0.0 with prefix length 8; therefore, the debug output shows up only for that prefix.

```
route-policy policy_b
  if destination in (10.0.0.0/8) then
    pass
  else
    drop

  endif
end-policy


debug bgp update  route-policy policy_b
```

BGP Attributes and Operators

This table summarizes the BGP attributes and operators per attach points.

Table 16: BGP Attributes and Operators

Attach Point	Attribute	Match	Set
aggregation	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is, ge, le, eq	—
	as-path-unique-length	is, ge, le, eq	—
	community	is-empty matches-any matches-every	set set additive delete in delete not in delete all
	destination	in	—
	extcommunity cost	—	set set additive
	local-preference	is, ge, le, eq	set
	med	is, eg, ge, le	setset +set -
	next-hop	in	set
	origin	is	set
	source	in	—
	suppress-route	—	suppress-route
	weight	—	set

Attach Point	Attribute	Match	Set
allocate-label	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is, ge, le, eq	—
	as-path-unique-length	is, ge, le, eq	—
	community	is-empty matches-any matches-every	—
	destination	in	—
	label	—	set
	local-preference	is, ge, le, eq	—
	med	is, eg, ge, le	—
	next-hop	in	—
	origin	is	—
	source	in	—
clear-policy	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is, ge, le, eq	—
	as-path-unique-length	is, ge, le, eq	—

Attach Point	Attribute	Match	Set
dampening	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is, ge, le, eq	—
	as-path-unique-length	is, ge, le, eq	—
	community	is-empty matches-any matches-every	—
	dampening	—/	set dampening
	destination	in	—
	local-preference	is, ge, le, eq	—
	med	is, eg, ge, le	—
	next-hop	in	—
	origin	is	—
	source	in	—
debug	destination	in	—
default originate	med	—	set set + set -
	rib-has-route	in	—

Attach Point	Attribute	Match	Set
neighbor-in	as-path	in is-local length NA neighbor-is originates-from passes-through unique-length	prepend prepend most-recent remove as-path private-as replace
	as-path-length	is, ge, le, eq	—
	as-path-unique-length	is, ge, le, eq	—
	communitycommunity with ‘peeras’	is-empty matches-any matches-every	set set additive delete-in delete-not-in delete-all
	destination	in	—
	extcommunity cost	—	set set additive
	extcommunity rt	is-empty matches-any matches-every matches-within	set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every matches-within	—
	local-preference	is, ge, le, eq	set
	med	is, eg, ge, le	set set + set -

Attach Point	Attribute	Match	Set
	next-hop	in	set set peer address
	origin	is	set
	route-aggregated	route-aggregated	NA
	source	in	—
	weight	—	set

Attach Point	Attribute	Match	Set
neighbor-out	as-path	in is-local length — neighbor-is originates-from passes-through unique-length	prepend prepend most-recent remove as-path private-as replace
	as-path-length	is, ge, le, eq	—
	as-path-unique-length	is, ge, le, eq	—
	communitycommunity with ‘peeras’	is-empty matches-any matches-every	set set additive delete-in delete-not-in delete-all
	destination	in	—
	extcommunity cost	—	set set additive
	extcommunity rt	is-empty matches-any matches-every matches-within	set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every matches-within	—
	local-preference	is, ge, le, eq	set
	med	is, eg, ge, le	

Attach Point	Attribute	Match	Set
			set set + set - set max-unreachable set igp-cost
	next-hop	in	set set self
	origin	is	set
	path-type	is	—
	rd	in	—
	route-aggregated	route-aggregated	—
	source	in	—
	unsuppress-route	—	unsuppress-route
	vpn-distinguisher	—	set
neighbor-orf	orf-prefix	in	n/a

Attach Point	Attribute	Match	Set
network	as-path	—	prepend
	community	—	set set additive delete-in delete-not-in delete-all
	destination	in	—
	extcommunity cost	—	set set additive
	mpls-label	route-has-label	—
	local-preference	—	set
	med	—	set set+ set-
	next-hop	in	set
	origin	—	set
	route-type	is	—
	tag	is, ge, le, eq	—
	weight	—	set
next-hop	destination	in	—
	protocol	is,in	—
	source	in	—

Attach Point	Attribute	Match	Set
redistribute	as-path	—	prepend
	community	—	set set additive delete in delete not in delete all
	destination	in	—
	extcommunity cost	—	setset additive
	local-preference	—	set
	med	—	set set+ set-
	next-hop	in	set
	origin	—	set
	mpls-label	route-has-label	—
	route-type	is	—
	tag	is, eq, ge, le	—
	weight	—	set
retain-rt	extcommunity rt	is-empty matches-any matches-every matches-within	—

Attach Point	Attribute	Match	Set
show	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is, ge, le, eq	—
	as-path-unique-length	is, ge, le, eq	—
	community	is-empty matches-any matches-every	—
	destination	in	—
	extcommunity rt	is-empty matches-any matches-every matches-within	—
	extcommunity soo	is-empty matches-any matches-every matches-within	—
	med	is, eg, ge, le	—
	next-hop	in	—
	origin	is	—
	source	in	—

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 17: Restricted BGP Operations by Attach Point

Command	import	export	aggregation	redistribution
prepend as-path most-recent	eBGP only	eBGP only	n/a	n/a
replace as-path	eBGP only	eBGP only	n/a	n/a
set med igp-cost	forbidden	eBGP only	forbidden	forbidden
set weight	n/a	forbidden	n/a	n/a
suppress	forbidden	forbidden	n/a	forbidden

Default-Information Originate

The default-information originate attach point allows the user to conditionally inject the default route 0.0.0.0/0 into the OSPF link-state database, which is done by evaluating the attached policy. If any routes in the local RIB pass the policy, then the default route is inserted into the link-state database.

The following example shows how to generate a default route if any of the routes that match 10.0.0.0/8 ge 8 le 25 are present in the RIB:

```
route-policy ospf-originate
  if rib-has-route in (10.0.0.0/8 ge 8 le 25) then
    pass
  endif
end-policy

router ospf 1
  default-information originate policy ospf-originate
  .
  .
  .
```

OSPF Policy Attach Points

This section describes each of the OSPF policy attach points and provides a summary of the OSPF attributes and operators.

Redistribute

The redistribute attach point within OSPF injects routes from other routing protocol sources into the OSPF link-state database, which is done by selecting the routes it wants to import from each protocol. It then sets the OSPF parameters of cost and metric type. The policy can control how the routes are injected into OSPF by using the **set metric-type** or **set ospf-metric** command.

The following example shows how to redistribute routes from IS-IS instance instance_10 into OSPF instance 1 using the policy OSPF-redist. The policy sets the metric type to type-2 for all redistributed routes. IS-IS routes with a tag of 10 have their cost set to 100, and IS-IS routes with a tag of 20 have their OSPF cost set

to 200. Any IS-IS routes not carrying a tag of either 10 or 20 are not be redistributed into the OSPF link-state database.

```
route-policy OSPF-redist
  set metric-type type-2
  if tag eq 10 then

 ..set ospf-metric  2
  elseif tag eq 20 then

 ..set ospf-metric   3
  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 11 .105.3.0/24, 11 .105.7.0/24, 11 .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 (11
.105.3.0/24, 11
.105.7.0/24, 11
.105.13.0/24) then
      pass
    endif
  end-policy

  router ospf 1
    area 1
      route-policy OSPF-area-out out
```

OSPF Attributes and Operators

This table summarizes the OSPF attributes and operators per attach points.

Table 18: OSPF Attributes and Operators

Attach Point	Attribute	Match	Set
default-information originate	ospf-metric	—	set
	metric-type	—	set
	tag	—	set
	rib-has-route	in	—
redistribute	destination	in	—
	metric-type	—	set
	ospf-metric	—	set
	next-hop	in	—
	mpls-label	route-has-label	—
	rib-metric	is, le, ge, eq	n/a
	route-type	is	—
	tag	is, eq, ge, le	set
area-in	destination	in	—
area-out	destination	in	—

Attach Point	Attribute	Match	Set
spf-prefix-priority	destination	in	n/a
	spf-priority	n/a	set
	tag	is, le, ge, eq	n/a

Distribute-list in

The distribute-list in attach point within OSPF allows use of route policies to filter OSPF prefixes. The distribute-list in route-policy can be configured at OSPF instance, area, and interface levels. The route-policy used in the distribute-list in command supports match statements, "destination" and "rib-metric". The "set" commands are not supported in the route-policy.

These are examples of valid route-policies for "distribute-list in":

```
route-policy DEST
  if destination in (10.10.10.10/32) then
    drop
  else
    pass
  endif
end-policy


route-policy METRIC
  if rib-metric ge 10 and rib-metric le 19 then
    drop
  else
    pass
  endif
end-policy


prefix-set R-PFX
  10.10.10.30
end-set

route-policy R-SET
  if destination in R-PFX and rib-metric le 20 then
    pass
  else
    drop
  endif
end-policy
```

OSPFv3 Policy Attach Points

This section describes each of the OSPFv3 policy attach points and provides a summary of the OSPFv3 attributes and operators.

Redistribute

The redistribute attach point within OSPFv3 injects routes from other routing protocol sources into the OSPFv3 link-state database, which is done by selecting the route types it wants to import from each protocol. It then

sets the OSPFv3 parameters of cost and metric type. The policy can control how the routes are injected into OSPFv3 by using the **metric type** command.

The following example shows how to redistribute routes from BGP instance 15 into OSPF instance 1 using the policy OSPFv3-redist. The policy sets the metric type to type-2 for all redistributed routes. BGP routes with a tag of 10 have their cost set to 100, and BGP routes with a tag of 20 have their OSPFv3 cost set to 200. Any BGP routes not carrying a tag of either 10 or 20 are not be redistributed into the OSPFv3 link-state database.

```
route-policy OSPFv3-redist
  set metric-type type-2
  if tag eq 10 then
    set extcommunity cost 100
  elseif tag eq 20 then
    set extcommunity cost 200
  else
    drop
  endif
end-policy

router ospfv3 1
  redistribute bgp 15 policy OSPFv3-redist
  .
  .
  .
```

OSPFv3 Attributes and Operators

This table summarizes the OSPFv3 attributes and operators per attach points.

Table 19: OSPFv3 Attributes and Operators

Attach Point	Attribute	Match	Set
default-information originate	ospf-metric	—	set
	metric-type	—	set
	tag	—	set
	rib-has-route	in	—
redistribute	destination	in	—
	ospf-metric	—	set
	metric-type	—	set
	route-type	is	—
	tag	is, eq, ge, le	—

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.

Default-Information Originate

The default-information originate attach point within IS-IS allows the default route 0.0.0.0/0 to be conditionally injected into the IS-IS route database.

The following example shows how to generate an IPv4 unicast default route if any of the routes that match 10.0.0.0/8 ge 8 le 25 is present in the RIB. The cost of the IS-IS route is set to 100 and the level is set to level-1-2 on the default route that is injected into the IS-IS database.

```
route-policy isis-originate
  if rib-has-route in (10.0.0.0/8 ge 8 le 25) then

    set level level-1-2
  endif
end-policy

router isis instance_10
  address-family ipv4 unicast
    default-information originate policy isis_originate
    .
```

Inter-area-propagate

The inter-area-propagate attach point within IS-IS allows the prefixes to be conditionally propagated from one level to another level within the same IS-IS instance.

The following example shows how to allow prefixes to be leaked from the level 1 LSP into the level 2 LSP if any of the prefixes match 10.0.0.0/8 ge 8 le 25.

```
route-policy isis-propagate
  if destination in (10.0.0.0/8 ge 8 le 25) then
    pass
  endif
end-policy


router isis instance_10
  address-family ipv4 unicast
    propagate level 1 into level 2 policy isis-propagate
    .
```

Nondestructive Editing of Routing Policy

The Nondestructive Editing of Routing Policy changes the default exit behavior under routing policy configuration mode to terminate the configuration.

The default **exit** command acts as end-policy, end-set, or end-if. If the **exit** command is executed under route policy configuration mode, the changes are applied and configuration is updated. This destructs the existing policy. The **rpl set-exit-as-abort** command allows to overwrite the default behavior of the **exit** command under the route policy configuration mode.

Attached Policy Modification

Policies that are in use do, on occasion, need to be modified. In the traditional configuration model, a policy modification would be done by completely removing the policy and reentering re-entering it. However, this model allows for a window of time in which no policy is attached and default actions to be used, which is an opportunity for inconsistencies to exist. To close this window of opportunity, you can modify a policy in use at an attach point by respecifying it, which allows for policies that are in use to be changed, without having a window of time in which no policy is applied at the given attach point.

Note A route policy or set that is in use at an attach point cannot be removed because this removal would result in an undefined reference. An attempt to remove a route policy or set that is in use at an attach point results in an error message to the user.

Nonattached Policy Modification

As long as a given policy is not attached at an attach point, the policy is allowed to refer to nonexistent sets and policies. Configurations can be built that reference sets or policy blocks that are not yet defined, and then later those undefined policies and sets can be filled in. This method of building configurations gives much greater flexibility in policy definition. Every piece of policy you want to reference while defining a policy need not exist in the configuration. Thus, you can define a policy sample1 that references a policy sample2 using an apply statement even if the policy sample2 does not exist. Similarly, you can enter a policy statement that refers to a nonexistent set.

However, the existence of all referenced policies and sets is enforced when a policy is attached. Thus, if a user attempts to attach the policy sample1 with the reference to an undefined policy sample2 at an inbound BGP policy using the statement **neighbor 1.2.3.4 address-family ipv4 unicast policy sample1 in**, the configuration attempt is rejected because the policy sample2 does not exist.

Editing Routing Policy Configuration Elements

RPL is based on statements rather than on lines. That is, within the begin-end pair that brackets policy statements from the CLI, a new line is merely a separator, the same as a space character.

The CLI provides the means to enter and delete route policy statements. RPL provides a means to edit the contents of the policy between the begin-end brackets, using a text editor. The following text editors are available on the software for editing RPL policies:

- Nano (default)
- Emacs

Editing Routing Policy Configuration Elements Using Emacs Editor

To edit the contents of a routing policy using the Emacs editor, use the following CLI command in XR EXEC mode:

```
edit

  route-policy
```

name

emacs

A copy of the route policy is copied to a temporary file and the editor is launched. After editing, save the editor buffer by using the Ctrl-X and Ctrl-S keystrokes. To save and exit the editor, use the Ctrl-X and Ctrl-C keystrokes. When you quit the editor, the buffer is committed. If there are no parse errors, the configuration is committed:

```
RP/0/RP0/CPU0:router# edit route-policy policy_A
----------------------------------------
== MicroEMACS 3.8b () == rpl_edit.139281 ==
  if destination in (2001::/8) then
    drop
  endif
end-policy
!

== MicroEMACS 3.8b () == rpl_edit.139281 ==
Parsing.
83 bytes parsed in 1 sec (82)bytes/sec
Committing.
1 items committed in 1 sec (0)items/sec
Updating.
Updated Commit database in 1 sec
```

If there are parse errors, you are asked whether editing should continue:

```
RP/0/RP0/CPU0:router#edit route-policy policy_B
== MicroEMACS 3.8b () == rpl_edit.141738
route-policy policy_B
 set metric-type type_1
 if destination in (2001::/8) then
    drop
  endif
end-policy
!
== MicroEMACS 3.8b () == rpl_edit.141738 ==
Parsing.
105 bytes parsed in 1 sec (103)bytes/sec

% Syntax/Authorization errors in one or more commands.!! CONFIGURATION
FAILED DUE TO SYNTAX/AUTHORIZATION ERRORS
 set metric-type type_1
 if destination in (2001::/8) then
    drop
  endif
end-policy
!

Continue editing? [no]:
```

If you answer **yes**, the editor continues on the text buffer from where you left off. If you answer **no**, the running configuration is not changed and the editing session is ended.

Editing Routing Policy Configuration Elements Using 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 mode.

Editing Routing Policy Configuration Elements Using Nano Editor

To edit the contents of a routing policy using the Nano editor, use the following CLI command in XR EXEC mode:

```
edit route-policy
  name
  nano
```

A copy of the route policy is copied to a temporary file and the editor is launched. After editing, enter Ctrl-X to save the file and exit the editor. The available editor commands are displayed on screen.

Detailed information on using the Nano editor is available at this URL: http://www.nano-editor.org/.

Not all Nano editor features are supported on the software.

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, the software 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 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 ChildA
 if destination in (10.10.0.0/16) then
  set local-pref 111
 endif
end-policy
!

route-policy ChildB
 if as-path originates-from '222' then
  set community (333:222) additive
 endif
end-policy
!

route-policy Parent
 if apply ChildA and apply ChildB 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 ChildA is pass */
  set local-pref 111
  set community (333:444) additive /*From else block */
 elseif as-path originates-from '222' then /*Only Policy ChildB is pass */
  set community (333:222, 333:444) additive /*From else block */
 else
  set community (333:444) additive /*From else block */
 endif
end-policy
```

Apply Conditions can be used with parameters and are supported on all attach points and on all clients. Hierarchical Apply Conditions can be used without any constraints on a cascaded level.

Existing route policy semantics can be expanded to include this Apply Condition:

```
Route-policy policy_name
 If apply policyA and apply policyB then
      Set med 100
 Else if not apply policyD then
      Set med 200
 Else
    Set med 300
 Endif
End-policy
```

Behavior of pass/drop/done RPL Statements for Simple Hierarchical Policies

This table describes the behavior of **pass**/**drop**/**done** RPL statements, with a possible sequence for executing the **done** statement for Simple Hierarchical Policies.

Route-policies with simple hierarchical policies	Possible done statement execution sequence	Behavior
pass	**pass** Continue_list	Marks the prefix as "acceptable" and continues with execution of continue_list statements.
drop	Stmts_list **drop**	Rejects the route immediately on hitting the **drop** statement and stops policy execution.
done	Stmts_list **done**	Accepts the route immediately on hitting the **done** statement and stops policy execution.
pass followed by **done**	**pass** Statement_list **done**	Exits immediately at the **done** statement with "accept route".
drop followed by **done**	**drop** Statement list **done**	This is an invalid scenario at execution point of time. Policy terminates execution at the **drop** statement itself, without going through the statement list or the **done** statement; the prefix will be rejected or dropped.

Behavior of pass/drop/done RPL Statements for Hierarchical Policy Conditions

This section describes the behavior of **pass**/**drop**/**done** RPL statements, with a possible sequence for executing the **done** statement for Hierarchical Policy Conditions.

Terminology for policy execution: "true-path", "false-path", and "continue-path".

```
Route-policy parent
  If apply hierarchical_policy_condition then
     TRUE-PATH         : if hierarchical_policy_condition  returns TRUE then this path will
 be executed.
  Else
     FALSE-PATH       : if hierarchical_policy_condition  returns FALSE then this path will
 be executed.
  End-if
  CONTINUE-PATH             : Irrespective of the TRUE/FALSE this path will be executed.
 End-policy
```

Hierarchical policy conditions	Possible done statement execution sequence	Behavior
pass	**pass** Continue_list	Marks the return value as "true" and continues execution within the same policy condition. If there is no statement after "**pass**", returns "true".
pass followed by **done**	**pass** or **set** action statement Stmt_list **done**	Marks the return value as "true" and continues execution till the **done** statement. Returns "true" to the apply policy condition to take "true-path".
done	Stmt_list without **pass** or **set** operation DONE	Returns " false". Condition takes "false-path".
drop	Stmt_list **drop** Stmt_list	The prefix is dropped or rejected.

Nested Wildcard Apply Policy

The hierarchical constructs of Routing Policy Language (RPL) allows one policy to refer to another policy. The referred or called policy is known as a child policy. The policy from which another policy is referred is called calling or parent policy. A calling or parent policy can nest multiple child policies for attachment to a common set of BGP neighbors. The nested wildcard apply policy allows wildcard (*) based apply nesting. The wildcard operation permits declaration of a generic apply statement that calls all policies that contain a specific defined set of alphanumeric characters, defined on the router.

A wildcard is specified by placing an asterisk (*) at the end of the policy name in an apply statement. Passing parameters to wildcard policy is not supported. The wildcard indicates that any value for that portion of the apply policy matches.

To illustrate nested wildcard apply policy, consider this policy hierarchy:

```
route-policy Nested_Wilcard
apply service_policy_customer*
end-policy

route-policy service_policy_customer_a
if destination in prfx_set_customer_a  then
set extcommunity rt (1:1) additive
endif
end-policy

route-policy service_policy_customer_b
if destination in prfx_set_customer_b then
set extcommunity rt (1:1) additive
endif
end-policy
```

```
route-policy service_policy_customer_c
if destination in prfx_set_customer_c then
set extcommunity rt (1:1) additive
endif
end-policy
```

Here, a single parent apply statement (apply service_policy_customer*) calls (inherits) all child polices that contain the identified character string "service_policy_customer". As each child policy is defined globally, the parent dynamically nests the child policies based on the policy name. The parent is configured once and inherits each child policy on demand. There is no direct association between the parent and the child policies beyond the wildcard match statement.

VRF Import Policy Enhancement

The VRF RPL based import policy feature provides the ability to perform import operation based solely on import route-policy, by matching on route-targets (RTs) and other criteria specified within the policy. No need to explicitly configure import RTs under global VRF-address family configuration mode. If the import RTs and import route-policy is already defined, then the routes will be imported from RTs configured under import RT and then follows the route-policy attached at import route-policy.

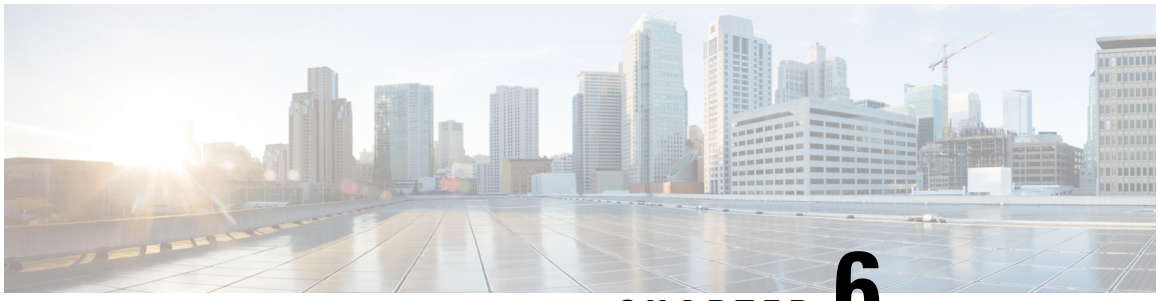
Use the **source rt import-policy** command under VRF sub-mode of VPN address-family configuration mode to enable this feature.

Match Aggregated Route

The Match Aggregated Route feature helps to match BGP aggregated route from the non-aggregated route. BGP can aggregate a group of routes into a single prefix before sending updates to a neighbor. With Match Aggregated Route feature, route policy separates this aggregated route from other routes.

Remove Private AS in Inbound Policy

BGP appends its own as-path before sending out packets to neighbors. When a packet traverses multiple iBGP neighbors, the as-path structure will have many private autonomous systems (AS) in them. The Remove Private AS in Inbound Policy will give the capability to delete those private autonomous systems using RPL route-policy. The **remove as-path private-as** command removes autonomous systems (AS) with AS number 64512 through 65535.



CHAPTER 6

Implementing RIP

The Routing Information Protocol (RIP) is a 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.

Table 20: Feature History Table

Feature Name	Release Information	Feature Description
RIPv2	Release 7.4.1	This feature enables RIP as the IGP of your network. RIP broadcasts UDP data packets to exchange routing information in networks that are flat rather than hierarchical, reducing network complexity and network management time.

- Prerequisites for Implementing RIP, on page 237
- Information About Implementing RIP , on page 237
- How to Implement RIP, on page 243
- Configuration Examples for Implementing RIP, on page 246
- Additional References, on page 248

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

While RIPv1 allows only contiguous blocks of hosts, subnets, or networks to be represented by a single route, RIPv2 allows Classless Inter-Domain Routing (CIDR). In addition, RIPv2 supports the following:

- Route summarization

- Variable-length subnet masks (VLSMs)
- Autonomous systems and the use of redistribution
- Multicast address 224.0.0.9 for RIP advertisements

The metric that RIP uses to rate the value of different routes is hop count. The hop count is the number of routers that can be traversed in a route. A directly connected network has a metric of zero; an unreachable network has a metric of 16. This small range of metrics makes RIP an unsuitable routing protocol for large networks.

Routing information updates are advertised every 30 seconds by default, and new updates discovered from neighbor routers are stored in a routing table.

Only RIP Version 2 (RIP v2), as specified in RFC 2453, is supported on Cisco IOS XR software and, by default, the software only sends and receives RIP v2 packets. However, you can configure the software to send, or receive, or both, only Version 1 packets or only Version 2 packets or both version type packets per interface.

Here are some good reasons to use RIP:

- Compatible with diverse network devices
- Best for small networks, because there is very little overhead, in terms of bandwidth used, configuration, and management time
- Support for legacy host systems

Because of RIP's ease of use, it is implemented in networks worldwide.

Note VRF does not allow configuration of a group applied directly under router RIP. A group can be configured if it is applied globally or under VRF.

Split Horizon for RIP

Normally, routers that are connected to broadcast-type IP networks and that use distance-vector routing protocols employ the split horizon mechanism to reduce the possibility of routing loops. Split horizon blocks information about routes from being advertised by a router out of any interface from which that information originated. This behavior usually optimizes communications among multiple routers, particularly when links are broken.

If an interface is configured with secondary IP addresses and split horizon is enabled, updates might not be sourced by every secondary address. One routing update is sourced per network number unless split horizon is disabled.

Note The split horizon feature is enabled by default. In general, we recommend that you do not change the default state of split horizon unless you are certain that your operation requires the change in order to properly advertise routes.

Route Timers for RIP

RIP uses several timers that determine such variables as the frequency of routing updates, the length of time before a route becomes invalid, and other parameters. You can adjust these timers to tune routing protocol performance to better suit your internetwork needs, by making the following timer adjustments to:

- The rate (time in seconds between updates) at which routing updates are sent
- The interval of time (in seconds) after which a route is declared invalid
- The interval (in seconds) during which routing information regarding better paths is suppressed
- The amount of time (in seconds) that must pass before a route is removed from the RIP topology table
- The amount of time delay between RIP update packets

The first four timer adjustments are configurable by the **timers basic** command. The **output-delay** command changes the amount of time delay between RIP update packets. See Customizing RIP section 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 21: Default Administrative Distances of Routing Protocols

Routing Protocols	Administrative Distance Value
Connected interface	0
Static route out an interface	0
Static route to next hop	1
EIGRP Summary Route	5
External BGP	20
Internal EIGRP	90
OSPF	110
IS-IS	115
RIP version 1 and 2	120
External EIGRP	170
Internal BGP	200
Unknown	255

An administrative distance is an integer from 0 to 255. In general, the higher the value, the lower the trust rating. An administrative distance of 255 means the routing information source cannot be trusted at all and should be ignored. Administrative distance values are subjective; there is no quantitative method for choosing them.

Default-Information Originate

The Default-Information Originate configuration allows the default route (0.0.0.0/0) to be advertised to peers in RIP. The receiving router accepts the default route, and then installs the route in the RIP database and the RIB table.

Starting from IOS XR Release 7.5.2, when a router is configured with the **default-information originate** command, the router ignores any received default routes from RIP peers. So, even if a default route is advertised to the router by a peer, the default route is not accepted or installed into the router's RIP database and the RIB table.

Example:

The following is a sample configuration:

```
router rip
 interface GigabitEthernet0/2/0/0
 !
 default-information originate
!
```

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.

Note The Cisco IOS XR software system security component implements various system security features including keychain management. For detailed information on keychain management concepts, configuration tasks, examples, and commands used to configure keychain management, refer the Implementing Keychain Management module in the System Security Configuration Guide, and the Keychain Management Commands module in the 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 be not be employed to secure it.

RIP employs two modes of authentication: keyed message digest mode and clear text mode. Use the **authentication keychain** *keychain-name* **mode** {**md5**|**text**} command to configure authentication using the keychain mechanism.

In cases where a keychain has been configured on RIP interface, but the keychain is actually not configured in the keychain database or keychain is not configured with MD5 cryptographic algorithm, all incoming RIP packets on the interface will be dropped. Outgoing packets will be sent without any authentication data.

In-bound RIP Traffic on an Interface

These are the verification criteria for all in-bound RIP packets on a RIP interface when the interface is configured with a keychain.

If ..	Then ..
The keychain configured on the RIP interface does not exist in the keychain database...	The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.
The keychain is not configured with a MD5 cryptographic algorithm...	The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.
The Address Family Identifier of the first (and only the first) entry in the message is not 0xFFFF, then authentication is not in use...	The packet will be dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.

If ..	Then ..
The MD5 digest in the ‘Authentication Data’ is found to be invalid...	The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.
Else, the packet is forwarded for the rest of the processing.	

Out-bound RIP Traffic on an Interface

These are the verification criteria for all out-bound RIP packets on a RIP interface when the interface is configured with a keychain.

If ..	Then ..
The MD5 digest in the ‘Authentication Data’ is found to be invalid...	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.

If ..	Then ..
The keychain is configured with a MD5 cryptographic algorithm...	The RIP packet passes authentication check at the remote/peer end, provided the remote router is also configured to authenticate the packets using the same keychain.
Else, RIP packets fail authentication check.	

How to Implement RIP

This section contains instructions for the following tasks:

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

Enabling RIP

This section 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.

These commands enable the RIP routing process and designates a neighbor router for exchanging RIP protocol information.

```
Router# configure
Router(config)# router rip
Router(config-rip)# neighbor 172.160.1.2
```

This command 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.

```
Router(config-rip)# broadcast-for-v2
```

These commands define the RIP routing protocol interface for accepting or sending packets that are RIP v1, RIP v2, or both RIP v1 and RIP v2.

```
Router(config-rip)# interface GigabitEthernet 0/1/0/0
Router(config-rip-if)# receive version 1 2
Router(config-rip-if)# send version 1 2
Router(config-rip-if)# commit
```

Customizing RIP

This section describes how to customize RIP for network timing and the acceptance of route entries.

```
Router# configure
Router(config)# router rip
```

(Optional) This command enables automatic route summarization of subnet routes into network-level routes. By default, auto-summary is disabled. If you have disconnected subnets, use the **no** keyword to disable

automatic route summarization and permit the software to send subnet and host routing information across classful network boundaries.

```
Router(config-rip)# auto-summary
```

(Optional) This adjusts RIP network timers. You can view the current and default timer values from the **show rip** command output.

```
Router(config-rip)# timers basic 5 15 15 30
```

(Optional) This changes the interpacket delay for the RIP updates sent. 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.

```
Router(config-rip)# output-delay 10
```

(Optional) This command configures NSF on RIP routes after a RIP process shutdown or restart.

```
Router(config-rip)# nsf
Router(config-rip)# interface GigabitEthernet 0/1/0/0
```

(Optional) This 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).

```
Router(config-rip-if)# metric-zero-accept
```

(Optional) This command disables the split horizon mechanism. By default, split horizon is enabled. In general, we do not recommend changing the state of the default for the **split-horizon** command, unless you are certain that your application requires a change to properly advertise routes. 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.

```
Router(config-rip-if)# split-horizon disable
```

Enables poison reverse processing of RIP router updates.

```
Router(config-rip-if)# poison-reverse
Router(config-rip-if)#  commit
```

Control Routing Information

This section 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.

This configures RIP and specifies a RIP neighbor.

```
Router# configure
Router(config)# router rip
```

```
Router(config-rip)# neighbor 172.160.1.2
Router(config-rip)# interface GigabitEthernet 0/1/0/0
```

(Optional) This command suppresses the sending of RIP updates on an interface, but not to explicitly configured neighbors.

```
Router(config-rip-if)# passive-interface
Router(config-rip-if)# exit
```

(Optional) This applies a routing policy to updates advertised to or received from a RIP neighbor.

```
Router(config-rip)# interface GigabitEthernet 0/2/0/0
Router(config-rip-if)# route-policy out
Router(config-rip-if)# commit
```

Creating a Route Policy for RIP

This task defines a route policy and shows how to apply it to a RIP instance. A route policy starts with the **route-policy** command, policy statements are added to the route policy, and it ends with the **end-policy** command. A route policy can control routes that are sent, received, redistributed, and control origination of the default route. A route policy is useful only when it is applied to a routing protocol's routes.

```
Router# configure
Router(config)# route-policy IN-IPv4
```

This command sets the RIP metric attribute.

```
Router(config-rpl)# set rip metric 42
Router(config-rpl)# end-policy
Router(config-rpl)# commit
```

This task applies a routing policy to updates advertised to or received from an RIP neighbor.

```
Router# configure
Router(config)# router rip
Router(config-rip)# route-policy rp1 in
Router(config-rip)# commit
```

Configuring RIP Authentication Keychain

Before you begin

All keychains need to be configured in Cisco IOS XR keychain database using configuration commands described in *Implementing Keychain Management* module of *System Security Configuration Guide* 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.

Configuring RIP Authentication Keychain for IPv4 Interface on a Non-default VRF

Configures Keyed message digest (**md5**) or clear text (**text**) authentication for RIP, for the specified VRF and interface.

```
Router(config)# router rip
Router(config-rip)# vrf vrf_rip_auth
Router(config-rip-vrf)# interface POS 0/6/0/0
Router(config-rip-if)# authentication keychain key1 mode md5
OR
Router(config-rip-if)# authentication keychain key1 mode text
Router(config-rip-if)# commit
```

Configuring RIP Authentication Keychain for IPv4 Interface on Default VRF

Configures Keyed message digest (**md5**) or clear text (**text**) authentication for RIP, for the default VRF and the specified interface.

```
Router# configure
Router(config)# router rip
Router(config-rip)# interface POS 0/6/0/0
Router(config-rip-if)# authentication keychain key1 mode md5
OR
Router(config-rip-if)# authentication keychain key1 mode text
Router(config-rip-if)# commit
```

Configuration Examples for Implementing RIP

This section provides the following configuration examples:

Configuring a Basic RIP Configuration: Example

The following example shows two Gigabit Ethernet interfaces configured with RIP.

```
interface GigabitEthernet0/6/0/0
ipv4 address 172.16.0.1 255.255.255.0
!
interface GigabitEthernet0/6/0/2
ipv4 address 172.16.2.12 255.255.255.0
!
router rip
interface GigabitEthernet0/6/0/0
!
interface GigabitEthernet0/6/0/2
!
```

Configuring RIP on the Provider Edge: Example

The following example shows how to configure basic RIP on the PE with two VPN routing and forwarding (VRF) instances.

```
router rip
interface GigabitEthernet0/6/0/0
!
vrf vpn0
interface GigabitEthernet0/6/0/2
!
!
vrf vpn1
interface GigabitEthernet0/6/0/3
!
```

Adjusting RIP Timers for each VRF Instance: Example

The following example shows how to adjust RIP timers for each VPN routing and forwarding (VRF) instance.

For VRF instance vpn0, the **timers basic** command sets updates to be broadcast every 10 seconds. If a router is not heard from in 30 seconds, the route is declared unusable. Further information is suppressed for an additional 30 seconds. At the end of the flush period (45 seconds), the route is flushed from the routing table.

For VRF instance vpn1, timers are adjusted differently: 20, 60, 60, and 70 seconds.

The **output-delay** command changes the interpacket delay for RIP updates to 10 milliseconds on vpn1. The default is that interpacket delay is turned off.

```
router rip
interface GigabitEthernet0/6/0/0
!
vrf vpn0
interface GigabitEthernet0/6/0/2
!
timers basic 10 30 30 45
!
vrf vpn1
interface GigabitEthernet0/6/0/3
!
timers basic 20 60 60 70
output-delay 10
!
```

Configuring Redistribution for RIP: Example

The following example shows how to redistribute Border Gateway Protocol (BGP) and static routes into RIP.

The RIP metric used for redistributed routes is determined by the route policy. If a route policy is not configured or the route policy does not set RIP metric, the metric is determined based on the redistributed protocol. For VPNv4 routes redistributed by BGP, the RIP metric set at the remote PE router is used, if valid.

In all other cases (BGP, IS-IS, OSPF, EIGRP, connected, static), the metric set by the **default-metric** command is used. If a valid metric cannot be determined, then redistribution does not happen.

```
route-policy ripred set rip-metric 5
end-policy
!

router rip vrf vpn0
interface GigabitEthernet0/6/0/2
!
redistribute connected default-metric 3
!
vrf vpn1
interface GigabitEthernet0/6/0/3
!
redistribute bgp 100 route-policy ripred redistribute static
default-metric 3
```

Configuring Route Policies for RIP: Example

The following example shows how to configure inbound and outbound route policies that are used to control which route updates are received by a RIP interface or sent out from a RIP interface.

```
prefix-set pf1 10.1.0.0/24
end-set
!
prefix-set pf2 150.10.1.0/24
end-set
!
route-policy policy_in
if destination in pf1 then pass
endif end-policy
!
route-policy pass-all pass
end-policy
!
route-policy infil
```

```
if destination in pf2 then add rip-metric 2
pass endif
end-policy
!
router rip
interface GigabitEthernet0/6/0/0 route-policy policy_in in
!
interface GigabitEthernet0/6/0/2
!
route-policy infil in route-policy pass-all out
```

Configuring Passive Interfaces and Explicit Neighbors for RIP: Example

The following example shows how to configure passive interfaces and explicit neighbors. When an interface is passive, it only accepts routing updates. In other words, no updates are sent out of an interface except to neighbors configured explicitly.

```
router rip
interface GigabitEthernet0/6/0/0 passive-interface
!
interface GigabitEthernet0/6/0/2
!
neighbor 172.17.0.1
neighbor 172.18.0.5
!
```

Controlling RIP Routes: Example

The following example shows how to use the **distance** command to install RIP routes in the Routing Information Base (RIB). The **maximum-paths** command controls the number of maximum paths allowed per RIP route.

```
router rip
interface GigabitEthernet0/6/0/0 route-policy polin in
!
distance 110
maximum-paths 8
!
```

Configuring RIP Authentication Keychain: Example

This example shows how to apply an authentication keychain on a RIP default VRF interface:

```
router rip
interface POS0/6/0/0
authentication keychain key1 mode md5
!
end
```

This example shows how to apply an authentication keychain on a RIP non-default interface:

```
router rip
vrf rip_keychain_vrf interface POS0/6/0/0
authentication keychain key1 mode md5
!
end
```

Additional References

The following sections provide references related to implementing RIP.

Related Documents

Related Topic	Document Title
RIP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Routing Command Reference Guide*
Site of Origin (SoO) support for RIP feature information	*Implementing MPLS Traffic Engineering on* module in the *MPLS Configuration Guide for Cisco NCS 540 Series Routers*
Cisco IOS XR getting started documentation	
Information about user groups and task IDs	*Configuring AAA Services on* module in the *System Security Configuration Guide for Cisco NCS 540 Series Routers*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

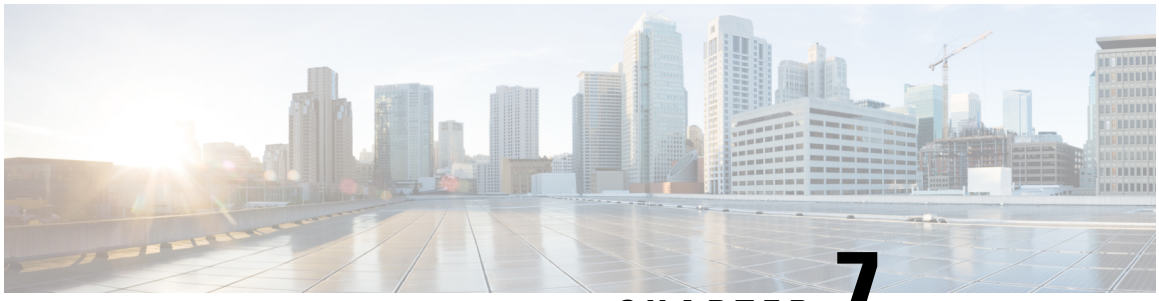
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

RFCs

RFCs	Title
RFC 2453	RIP Version 2

Technical Assistance

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



CHAPTER 7

Implementing 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 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.

References for Static Routes, on page 259 provides additional conceptual information on static routes.

This module describes how to implement static routes.

- Restrictions for Implementing Static Routes, on page 251
- Configure Static Route, on page 251
- Floating Static Routes , on page 253
- Configure Static Routes Between PE-CE Routers, on page 254
- IPv4 Multicast Static Routes, on page 256
- Default VRF, on page 258
- References for Static Routes, on page 259

Restrictions for Implementing Static Routes

These restrictions apply while implementing Static Routes:

- Static routing to an indirect next hop, (any prefix learnt through the RIB and may be more specific over the AIB), that is part of a local subnet requires configuring static routes in the global table indicating the egress interfaces as next hop. To avoid forward drop, configure static routes in the global table indicating the next-hop IP address to be the next hop.
- Generally, a route is learnt from the AIB in the global table and is installed in the FIB. However, this behavior will not be replicated to leaked prefixes. This could lead to inconsistencies in forwarding behavior.

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.

This task explains how to configure a static route.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router static**

Example:

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

Enters static route configuration mode.

Step 3 **vrf** *vrf-name*

Example:

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

(Optional) Enters VRF configuration mode.

If a VRF is not specified, the static route is configured under the default VRF.

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

Example:

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

Enters address family mode.

Step 5 *prefix mask* [**vrf** *vrf-name*] { *ip-address* | *interface-type* *interface-instance* } [*distance*] [**description** *text*] [**tag** *tag*] [**permanent**]

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 10.0.0.0/8 172.20.16.6 110
```

Configures an administrative distance of 110.

- This example shows how to route packets for network 10.0.0.0 through to a next hop at 172.20.16.6 if dynamic information with administrative distance less than 110 is not available.

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

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

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.

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

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
```

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.

Note By default, static routes have smaller administrative distances than dynamic routes, so static routes are preferred to dynamic routes.

Configure Floating Static Route

This task explains how to configure a floating static route.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router static**

Example:

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

Enters static route configuration mode.

Step 3 **vrf** *vrf-name*

Example:

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

(Optional) Enters VRF configuration mode.

If a VRF is not specified, the static route is configured under the default VRF.

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

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf)# address-family ipv6 unicast
```

Enters address family mode.

Step 5 *prefix mask* [**vrf** *vrf-name*] { *ip-address* | *interface-type* *interface-instance* } [*distance*] [**description** *text*] [**tag** *tag*] [**permanent**]

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
```

Configures an administrative distance of 201.

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

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

end —Prompts user to take one of these actions:

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

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).

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router static**

Example:

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

Enters static route configuration mode.

Step 3 **vrf** *vrf-name*

Example:

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

(Optional) Enters VRF configuration mode.

If a VRF is not specified, the static route is configured under the default VRF.

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

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf)# address-family ipv6 unicast
```

Enters address family mode.

Step 5 *prefix mask* [**vrf** *vrf-name*] { *ip-address* | *interface-type interface- path-id* } [*distance*] [**description** *text*] [**tag** *tag*] [**permanent**]

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
```

Configures an administrative distance of 201.

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

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

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.

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

In the following example, a static route between PE and CE routers is configured, and a VRF is associated with the static route:

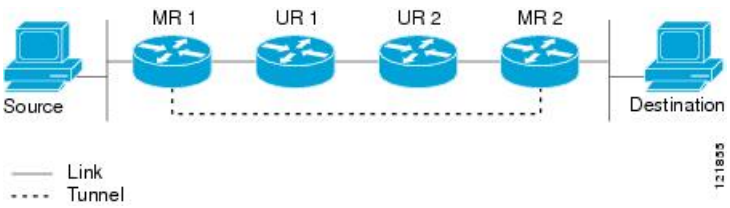
```
configure
 router static
  vrf vrf_A
  address-family ipv4 unicast
  0.0.0.0/0 2.6.0.2 120
  end
```

IPv4 Multicast Static Routes

IP multicast static routes (mroutes) allow you to have multicast paths diverge from the unicast paths. When using Protocol Independent Multicast (PIM), the router expects to receive packets on the same interface where it sends unicast packets back to the source. This expectation is beneficial if your multicast and unicast topologies are congruent. However, you might want unicast packets to take one path and multicast packets to take another.

The most common reason for using separate unicast and multicast paths is tunneling. When a path between a source and a destination does not support multicast routing, configuring two routers with a GRE tunnel between them is the solution. In the figure below, each unicast router (UR) supports unicast packets only; each multicast router (MR) supports multicast packets.

Figure 6: Tunnel for Multicast Packets



In the figure, the source delivers multicast packets to destination by using MR 1 and MR 2. MR 2 accepts the multicast packet only if it predicts it can reach source over the tunnel. If this situation is true, when the destination sends unicast packets to the source, MR 2 sends them over the tunnel. The check that MR2 can reach the source over the tunnel is a Reverse Path Forwarding (RPF) check, and the static mroute allows the check to be successful when the interface, on which the multicast packet arrives, is not the unicast path back to the source. Sending the packet over the tunnel could be slower than natively sending it through UR 2, UR 1, and MR 1.

A multicast static route allows you to use the configuration in the above figure by configuring a static multicast source. The system uses the configuration information instead of the unicast routing table to route the traffic. Therefore, multicast packets can use the tunnel without having the unicast packets use the tunnel. Static mroutes are local to the router they are configured on and not advertised or redistributed in any way to any other router.

Configure Multicast Static Routes

The following example shows how to configure multiple static routes in IPv4 and IPv6 address family configuration modes:

```
/* Enables a static routing process */
Router(config)# router static

/* Configures the IPv4 address-family for the unicast topology with a destination prefix.
*/
Router(config-static)# address-family ipv4 unicast
Router(config-static-afi)# 10.1.1.0/24 198.51.100.1
Router(config-static-afi)# 223.255.254.254/32 203.0.113.1
Router(config-static-afi)# exit

/* Configures the IPv4 address-family for the multicast topology with a destination prefix.
 */
Router(config-static)# address-family ipv4 multicast
Router(config-static-afi)# 198.51.100.20/32 209.165.201.0
Router(config-static-afi)# 192.0.2.10/32 209.165.201.0
Router(config-static-afi)# exit

/* Enable the address family IPv4 and IPv6 multicast on the next hop interface. */
Router(config)# interface TenGigE 0/0/0/0
Router(config-if)# address-family ipv4 multicast
Router(config-if)# address-family ipv6 multicast
```

Running Configuration

```
router static
 address-family ipv4 unicast
  10.1.1.0/24 198.51.100.1
  223.255.254.254/32 203.0.113.1
  !
address-family ipv4 multicast
  198.51.100.20/32 209.165.201.0
  192.0.2.10/32 209.165.201.0
  !
interface TenGigE 0/0/0/0
  address-family ipv4 multicast
  address-family ipv6 multicast
```

Verification

Verify the IPv4 multicast routes.

```
show route ipv4 multicast

Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
      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, G  - DAGR, l - LISP
       A - access/subscriber, a - Application route
       M - mobile route, r - RPL, t - Traffic Engineering, (!) - FRR Backup path

Gateway of last resort is 10.1.1.20 to network 0.0.0.0
```

```
i*L1 0.0.0.0/0 [115/10] via 10.1.1.20, 00:41:12, TenGigE0/0/0/6
C    10.1.1.0/24 is directly connected, 00:41:12, TenGigE0/0/0/0
L    10.1.1.10/32 is directly connected, 00:41:12, TenGigE0/0/0/0
S    172.16.2.10/32 [1/0] via 198.51.100.20, 00:41:12
i L1 172.16.3.1/32 [115/20] via 198.51.100.20, 00:41:12, TenGigE0/0/0/12
i L1 192.0.2.1/24 [115/20] via 198.51.100.20, 00:41:12, TenGigE0/0/0/1
```

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.

Note 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.

Associate VRF with a Static Route

This task explains how to associate a VRF with a static route.

Procedure

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **router static**

Example:

```
RP/0/
/CPU0:router(config)# router static
```

Enters static route configuration mode.

Step 3 **vrf** *vrf-name*

Example:

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

Enters VRF configuration mode.

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

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf)# address-family ipv6 unicast
```

Enters address family mode.

Step 5 *prefix mask* [**vrf** *vrf-name*] {**next-hop** *ip-address* | *interface-name* } {*path-id* } [*distance*] [**description** *text*] [**tag** *tag*] [**permanent**]

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
```

Configures an administrative distance of 201.

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

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

end —Prompts user to take one of these actions:

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

References for Static Routes

The following topics provide additional conceptual information on static routes:

- Static Route Functional Overview, on page 259
- Default Administrative Distance, on page 260
- Directly Connected Routes, on page 260
- Floating Static Routes , on page 253
- Fully Specified Static Routes , on page 261
- Recursive Static Routes , on page 261

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 TenGigE 0/0/0/0:

```
RP/0/RP0/CPU0:router(config)# router static
RP/0/RP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-static-afi)# 2001:0DB8::/32 TenGigE 0/0/0/0
```

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.

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.

Note By default, static routes have smaller administrative distances than dynamic routes, so static routes are preferred to dynamic routes.

Fully Specified Static Routes

In a fully specified static route, both the output interface and next hop are specified. This form of static route is used when the output interface is multiaccess and it is necessary to explicitly identify the next hop. The next hop must be directly attached to the specified output interface. The following example shows a definition of a fully specified static route:

```
RP/0/RP0/CPU0:router(config)# router static
RP/0/RP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-static-afi)# 2001:0DB8::/32 TenGigE 0/0/0/0 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.

Recursive Static Routes

In a recursive static route, only the next hop is specified. The output interface is derived from the next hop. The following example shows how to specify that all destinations with address prefix 2001:0DB8::/32 are reachable through the host with address 2001:0DB8:3000::1:

```
RP/0/RP0/CPU0:router(config)# router static
RP/0/RP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-static-afi)# 2001:0DB8::/32 2001:0DB8:3000::1
```

A recursive static route is valid (that is, it is a candidate for insertion in the routing table) only when the specified next hop resolves, either directly or indirectly, to a valid output interface, provided the route does not self-recurse, and the recursion depth does not exceed the maximum IPv6 forwarding recursion depth.

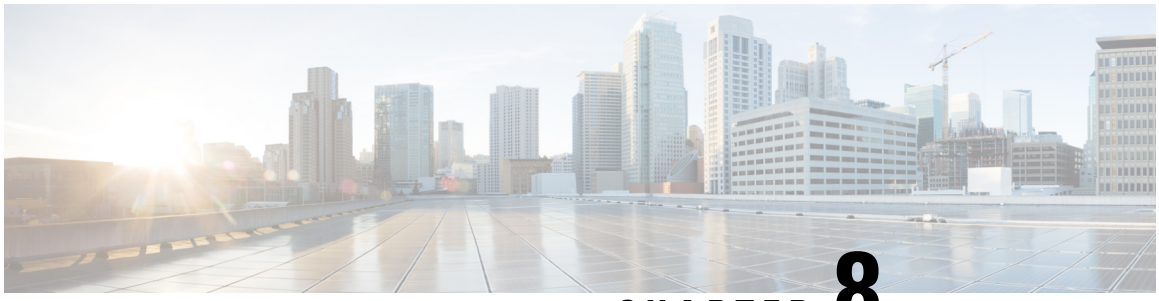
A route self-recurses if it is itself used to resolve its own next hop. If a static route becomes self-recursive, RIB sends a notification to static routes to withdraw the recursive route.

Assuming a BGP route 2001:0DB8:3000::0/16 with next hop of 2001:0DB8::0104, the following static route would not be inserted into the IPv6 RIB because the BGP route next hop resolves through the static route and the static route resolves through the BGP route making it self-recursive:

```
RP/0/RP0/CPU0:router(config)# router static
RP/0/RP0/CPU0:router(config-static)# address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-static-afi)# 001:0DB8::/32 2001:0DB8:3000::1
```

This static route is not inserted into the IPv6 routing table because it is self-recursive. The next hop of the static route, 2001:0DB8:3000:1, resolves through the BGP route 2001:0DB8:3000:0/16, which is itself a recursive route (that is, it only specifies a next hop). The next hop of the BGP route, 2001:0DB8::0104, resolves through the static route. Therefore, the static route would be used to resolve its own next hop.

It is not normally useful to manually configure a self-recursive static route, although it is not prohibited. However, a recursive static route that has been inserted in the routing table may become self-recursive as a result of some transient change in the network learned through a dynamic routing protocol. If this occurs, the fact that the static route has become self-recursive will be detected and it will be removed from the routing table, although not from the configuration. A subsequent network change may cause the static route to no longer be self-recursive, in which case it is re-inserted in the routing table.



CHAPTER 8

Implementing BFD

- BFD Overview , on page 263
- BFD Session Types, on page 270
- BFD Singlepath Sessions, on page 270
- BFD Multipath Sessions, on page 287
- Seamless Bidirectional Forwarding Detection, on page 294
- Coexistence Between BFD over Bundle and BFD over Logical Bundle , on page 298
- BFD CPU Offload Support for IPv6, on page 304
- BFD Object Tracking, on page 306
- BFD Dampening, on page 307

BFD Overview

Bidirectional Forwarding Detection (BFD) provides low-overhead, short-duration detection of failures in the path between adjacent routers. 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.

Tip You can programmatically configure BFD and retrieve operational data using `openconfig-bfd.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide*.

Features Unsupported

- BFD echo mode and encryption are not supported.
- BFD over MPLS tunnel interfaces is not supported.
- Dampening extensions for BFD are not supported.
- BFD down dampening is not supported.
- BFD IPv6 Dampening is not supported.
- SNMP traps are not supported for multipath BFD sessions.
- BFD Over GRE is not supported.

- BFD over PWHE is not supported.
- Seamless BFD is not supported.
- BFD over Satellite interface is not supported.
- BFD Authentication is not supported.

Supported Functionalities

- BFD hardware offload is supported for both IPv4 and IPv6.
- BFD is only supported in IP core. It cannot coexist with Label distribution Protocol, or Segment Routing, or Traffic Engineering in the core. This is applicable for releases prior to IOS XR Release 7.1.1.
- BFD over Bundle (BoB) over IPv6 is not supported with dyamically configured link-local address. It must be statically configured.
- Dampening extensions for BFD are not supported.
- Egress IPv4 ACLs block all traffic, including router-generated traffic for the following routers and line cards:
  - NC57-24DD
  - NC57-18DD-SE
  - NC57-36H-SE
  - NC57-36H6D-S
  - NC57-MOD-S
  - NCS-57B1-6D24-SYS
  - NCS-57B1-5DSE-SYS

  For all other routers and line cards, egress IPv4 ACLs do not block certain router-generated traffic, such as ICMP messages.

Feature Limitations

- Egress ACL with drop rule for src-ip equal to 0.0.0.0 will drop BFD-V4 Tx packets on that interface. This is because, BFD-V4 packets generated by OAMP will have src.ip 0.0.0.0 due to its limitation. And the actual source IP value is filled in ETPP block in pipeline before sending the packet. Since egress ACL is applied before ETPP, the BFD packets are dropped.
- BFD over Bundle (BOB) uses two entries in the MAC address table for each of IPv4 and IPv6, regardless of the number of BOB sessions on the Network Processing Unit (NPU). This MAC table is shared with other features such as Bridge Virtual Interface (BVI), Hot Standby Router Protocol (HSRP), Virtual Router Redundancy Protocol (VRRP), and DHCP, all of which also consume entries from the same MAC table. The MAC table has a limited capacity of 16 entries. If the MAC address for BOB cannot be programmed because the table is full with other MAC entries, the BOB session will fail to establish. This limitation means that the MAC table resource must be managed carefully to ensure successful BOB session establishment.
- BOB feature is supported only in IETF mode.

- If peer routers have mismatched BFD configurations, with one router configured with **bundle coexistence bob-blb logical** command and the other not, the BFD session may behave unexpectedly.

Note The **bundle coexistence bob-blb logical** command is specific to Cisco IOS XR. Therefore, for consistent BFD behaviour, this command must be configured identically across all peer nodes.

- The **bfd interface** *type interface-path-id* **disable** command is not supported.

BFD Timers

The BFD timers are applicable on the following NCS 540 variants:

Medium Density XR NCS 540 - N540-24Z8Q2C-SYS, N540-28Z4C-SYS, N540X-ACC-SYS, N540-ACC-SYS

Medium Density XR NCS 540 - N540-28Z4C-SYS-A, N540-28Z4C-SYS-D, N540X-16Z4G8Q2C-A, N540X-16Z4G8Q2C-D, N540X-16Z8Q2C-D, N540-12Z20G-SYS-A,N540-12Z20G-SYS-D, N540X-12Z16G-SYS-A, N540X-12Z16G-SYS-D

Small Density XR NCS 540 - N540X-6Z18G-SYS-A, N540X-6Z18G-SYS-D, N540X-8Z16G-SYS-A, N540X-8Z16G-SYS-D

Note If the timer is configured below the minimum timer supported, some undesirable behavior can be seen in BFD. customers some time will configure 3 msec as timer and will miss the minimum timer of 4 msec.

Note The router uses six unique timer profiles. Four timers profiles are available when you configure BFD over Bundle (BoB). Up to five timers profiles are available when you configure BoB.

Table 22: IPv4 BFD Timers

Type of BFD Session	Minimum Timer Supported	Minimum Multipliers Value	Supported Minimum-Interval Value (Up to 6 Unique Timers Profiles)
Single Hop	4ms	3	Any
BFD over Bundle Members (BoB)	4ms	3	Any
BFD over Logical bundle (BLB)	100ms (starting Release 24.3.1) 300ms (prior to Release 24.3.1)	3	Any

Type of BFD Session	Minimum Timer Supported	Minimum Multipliers Value	Supported Minimum-Interval Value (Up to 6 Unique Timers Profiles)
BGP Multi Hop	50ms	3	Any
BFD Over BVI	50ms	3	Any

Table 23: IPv6 BFD Timers

Type of BFD Session	Minimum Timer Supported	Minimum Multipliers Value	Supported Timer Profile (Up to 6 unique timer profiles)	Maximum Scale depending on Minimum Interval
Single Hop	4ms	3	Any	150 (with 8ms and above, all 256 sessions are configurable)
BFD over Bundle Members (BoB)	4ms	3	Any	150ms (with 8ms and above, all 256 sessions are configurable)
BFD over Logical bundle (BLB)	100ms (starting Release 24.3.1) 300ms (prior to Release 24.3.1)	3	Any	256
BGP Multi Hop	50ms	3	Any	256
BFD Over BVI	50ms	3	Any	250 or Max MP scale- whichever is lower

Enable and Disable IPv6 Checksum Calculations for BFD on a Router

Perform the following steps to configure IPv6 checksum calculations for BFD on a Router.

```
RP/0/RP0/CPU0:router(config)# bfd
RP/0/RP0/CPU0:router(config-bfd-if)# ipv6 checksum disable
RP/0/RP0/CPU0:router(config-bfd-if)# commit
```

Configure BFD Under a Dynamic Routing Protocol or Use a Static Route

To establish a BFD neighbor, complete at least one of the following procedures to configure BFD under a dynamic routing protocol or to use a static route:

Enable BFD for OSPF on an Interface

Perform the following steps 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; only the command mode differs.

Note BFD per interface configuration is supported for OSPF and IS-IS only.

```
Router# configure

/* Enter OSPF configuration mode to configure the OSPF routing process. */
Router(config)# router ospf 0

/* Set the BFD minimum interval. The range is from 15 to 30000 milliseconds. */
Router(config-ospf)# bfd minimum-interval 6500

/* Set the BFD multiplier. */
Router(config-ospf)# bfd multiplier 7

/* Configure an Open Shortest Path First (OSPF) area. */
Router(config-ospf)# area 0

/* Enter interface configuration mode. */
Router(config-ospf-ar)# interface gigabitEthernet 0/3/0/1

/* Enable BFD to detect failures in the path between adjacent forwarding engines. */
Router(config-ospf-ar-if)# bfd fast-detect
```

Running Configuration

```
configure
  router ospf 0
  bfd minimum-interval 6500
  bfd multiplier 7
   area 0
    interface gigabitEthernet 0/3/0/1
     bfd fast-detect
```

Verification

Verify that BFD is enabled on the appropriate interface.

```
Router(config-ospf-ar-if)# show run router ospf

 router ospf 0
 bfd minimum-interval 6500
 bfd multiplier 7
 area 0
 interface gigabitEthernet 0/3/0/1
 bfd fast-detect

/* Verify the details of the IPv4 BFD session in the source router. */

Router# show bfd session

Interface  Dest Addr  Local det time(int*mult)  State  Echo  Async   H/W   NPU
---------  ---------  --------  -------------   -----  ----  -----   ---   ---
```

```
Te0/0/0/0  10.23.1.2  0s(0s*0)  300ms(100ms*3)  UP     Yes               0/RP0/CPU0
BE3739     10.23.1.2  n/a       n/a             UP     No    n/a
```

Enable BFD over BGP

Perform the following steps to configure BFD over BGP. The following example shows how to configure BFD between autonomous system 65000 and neighbor 192.168.70.2:

```
Router# configure
Router(config)# router bgp 65000
Router(config-bgp)# bfd multiplier 2
Router(config-bgp)# bfd minimum-interval 20
Router(config-bgp)# neighbor 192.168.70.24
Router(config-bgp-nbr)# remote-as 2
Router(config-bgp-nbr)# bfd fast-detect
Router(config-bgp-nbr)# commit
Router(config-bgp-nbr)# end
```

Running Configuration

```
 router bgp 65000
  bfd multiplier 2
  bfd minimum-interval 20
  neighbor 192.168.70.24
  remote-as 2
  bfd fast-detect
  commit
  end
```

Verification

Verify that BFD has been enabled over BGP.

```
Router# show run router bgp
router bgp 65000
  bfd multiplier 2
  bfd minimum-interval 20
  neighbor 192.168.70.24
  remote-as 2
  bfd fast-detect
```

Enable BFD on an IPv4 Static Route

The following procedure shows how to enable BFD on an IPv4 static route.

```
RP/0/RSP0/CPU0:router# configure

/*Enter static route configuration mode to configure static routing. */
RP/0/RSP0/CPU0:router(config)# router static

/* Enable BFD fast-detection on the specified IPV4 unicast destination address prefix and
on the forwarding next-hop address.*/
RP/0/RSP0/CPU0:router(config-static)# address-family ipv4 unicast 10.2.2.0/24 10.6.0.1 bfd
 fast-detect minimum-interval 1000 multiplier 5

RP/0/RSP0/CPU0:router(config-static)# commit
```

Running Configuration

```
configure
 router static
  address-family ipv4 unicast 10.2.2.0/24 10.6.0.1 bfd fast-detect minimum-interval 1000
multiplier 5
  commit
```

Verification

Verify that BFD is enabled on the appropriate interface.

```
RP/0/RSP0/CPU0:router# show run router static address-family ipv4 unicast

router static
 address-family ipv4 unicast
  10.2.2.0/24 10.6.0.1 bfd fast-detect minimum-interval 1000 multiplier 5
   commit
 !
!
```

Enable BFD on an IPv6 Static Route

The following procedure describes how to enable BFD on a IPv6 static route.

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

/* Enter static route configuration mode to configure static routing.  */
RP/0/RP0/CPU0:router(config)# router static

/* Enable BFD fast-detection on the specified IPv6 unicast destination address prefix and
on the forwarding next-hop address. */
/* BFD sessions are established with the next hop 2001:0DB8:D987:398:AE3:B39:333:783 when
it becomes reachable. */

RP/0/RP0/CPU0:router(config-static)# address-family ipv6 unicast  2001:0DB8:C18:2:1::F/64
2001:0DB8:D987:398:AE3:B39:333:783 bfd fast-detect minimum-interval 150 multiplier 4

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

Running Configuration

```
configure
 router static
  address-family ipv6 unicast 2001:0DB8:C18:2:1::F/64 2001:0DB8:D987:398:AE3:B39:333:783
bfd fast-detect minimum-interval 150 multiplier 4
  commit
```

Verification

Verify that BFD is enabled on the appropriate interface.

```
RP/0/RP0/CPU0:router# show run router static address-family ipv6 unicast

 configure
 router static
 address-family ipv6 unicast 2001:0DB8:C18:2:1::F/64 2001:0DB8:D987:398:AE3:B39:333:783 bfd
 fast-detect minimum-interval 150 multiplier 4
  commit
```

Clear and Display 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.

```
RP/0/RP0/CPU0:router# show bfd counters all packet location 0/3/cpu0
RP/0/RP0/CPU0:router# clear bfd counters all packet location 0/3/cpu0
RP/0/RP0/CPU0:router# show bfd counters all packet location 0/3/cpu0
```

BFD Session Types

There are two types of BFD sessions:

- Single Path Sessions
- Multipath Sessions

BFD Singlepath Sessions

BFD over Bundle

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

The BoB feature enables BFD sessions to monitor the status of individual bundle member links. BFD notifies the bundle manager immediately when one of the member links goes down, and reduces the bandwidth used by the bundle.

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

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

Limitations and Restrictions for BFD over Bundle

The following are the limitations and restrictions in using BoB feature:

- Each BoB session must use a unique destination IP address. If multiple BoB sessions across different bundles share the same destination IP address, BFD should not be enabled.
- It is only supported in IETF mode.
- It is only supported on the main bundle interface; it is not supported on bundle subinterfaces.
- It is not supported on routing protocols, such as OSPF, ISIS, and BGP.
- When the BFD timer is configured to 4 ms, which is the most aggressive timer, 256 sessions can be brought up.

- BFD echo mode and encryption is not supported.

Configure BFD over Bundle

Configuring BFD over bundle involves the following steps:

- Specify the mode, BFD packet transmission intervals, and failure detection times on a bundle.

Note Repeat the same configuration steps in the destination router.

```
/* Enable and Disable IPv6 checksum calculations for BFD on a router. */

Router(config-if)# bfd
Router(config-bfd-if)# dampening disable
Router(config-bfd-if)# commit

/* Specify the mode, BFD packet transmission intervals, and failure detection times on a
bundle */

Router(config)# interface Bundle-Ether 3739
Router(config-if)# bfd mode ietf
Router(config-if)# bfd address-family ipv4 multiplier 3
Router(config-if)# bfd address-family ipv4 destination 10.23.1.2
Router(config-if)# bfd address-family ipv4 fast-detect
Router(config-if)# bfd address-family ipv4 minimum-interval 100
Router(config-if)# bfd address-family ipv6 multiplier 3
Router(config-if)# bfd address-family ipv6 destination 2001:DB8:1::2
Router(config-if)# bfd address-family ipv6 fast-detect
Router(config-if)# bfd address-family ipv6 minimum-interval 100
Router(config-if)# ipv4 address 10.23.1.1 255.255.255.252
Router(config-if)# ipv6 address 2001:DB8:1::2/120
Router(config-if)# load-interval 30
Router(config-if)# commit
Router(config)# interface TenGigE 0/0/0/0
Router(config-if)# bundle id 3739 mode active
```

Running Configuration

```
bfd
 dampening disable!
 !

interface Bundle-Ether3739
 bfd mode ietf
 bfd address-family ipv4 multiplier 3
 bfd address-family ipv4 destination 10.23.1.2
 bfd address-family ipv4 fast-detect
 bfd address-family ipv4 minimum-interval 100
 bfd address-family ipv6 multiplier 3
 bfd address-family ipv6 destination 2001:DB8:1::2
 bfd address-family ipv6 fast-detect
 bfd address-family ipv6 minimum-interval 100
 ipv4 address 10.23.1.1 255.255.255.252
 ipv6 address 2001:DB8:1::2/120
 load-interval 30
```

```
 !

interface TenGigE 0/0/0/0
 bundle id 3739 mode active
```

Verification

The following show command outputs displays the status of BFD sessions on bundle members:

```
/* Verify the details of the IPv4 BFD session. */

Router# show bfd all session

Interface   Dest Addr   Local det  time(int*mult) State   Echo   Async       H/W   NPU
----------  ----------  --------   -------------- -----  ----   -----       ---   ---
Te0/5/0/6   10.10.10.1   0s         450ms(150ms*3)  UP    Yes    0/RP0/CPU0
Te0/5/0/6   10.10.10.1   0s(0s*0)   450ms(150ms*3)  UP    Yes    0/RP1/CPU0
BE5         10.10.10.1   n/a        n/a             UP    No     n/a


/* Verify the details of the IPv6 BFD session. */
Router# show bfd all session

Interface    Dest Addr   Local det  time(int*mult)  State   Echo     Async      H/W   NPU
----------   ----------  --------   --------------  -----   ----     ---------  ---   ---
Te0/5/0/6    10:10::10:1   0s       450ms(150ms*3)   UP      Yes    0/RP0/CPU0
Te0/5/0/6    10:10::10:1 0s(0s*0)   450ms(150ms*3)   UP      Yes    0/RP1/CPU0
BE5          10:10::10:1   n/a        n/a            UP       No    n/a
```

Configuration Example

Enter the OSPF configuration mode to configure the OSPF routing process.

```
Router(config)# router ospf 0
Router(config)# configure LDP-IGP synchronization
Router(config)# mpls ldp sync
Router(config)# enable LDP auto-configuration for a specified OSPF instance
Router(config)# mpls ldp auto-config
```

Configure a BFD over Bundle Session on an Unnumbered Interface.

```
Router(config)# interface Bundle-Ether1
```

Set the BFD minimum interval.

```
Router(config)# bfd minimum-interval 999
```

Enables BFD between the local networking devices and the neighbor whose IP address you configured to be a BGP peer.

```
Router(config)# bfd fast-detect
```

Set the BFD multiplier.

```
Router(config)# bfd multiplier 3
```

Running Configuration

```
!
router ospf 1
log adjacency changes
router-id 10.0.0.4
mpls ldp sync
mpls ldp auto-config
```

```
prefix-suppression
auto-cost reference-bandwidth 1000000
area 0
  interface Bundle-Ether1
   bfd minimum-interval 999
   bfd fast-detect
   bfd multiplier 3
   network point-to-point
  !
!
bfd
multipath include location 0/RP0/CPU0
!
```

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.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer. This example configures the IP address 172.168.40.24 as a BGP peer.
Step 4	**remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system. This example configures the remote autonomous system to be 2002.
Step 5	**bfd fast-detect** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# bfd fast-detect`	Enables BFD between the local networking devices and the neighbor whose IP address you configured to be a BGP peer in Step 3. In the example in Step 3, the IP address 172.168.40.24 was set up as the BGP peer. In this example, BFD is enabled between the local

	Command or Action	Purpose
		networking devices and the neighbor 172.168.40.24.
Step 6	**bfd minimum-interval** *milliseconds* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)#bfd minimum-interval 6500`	Sets the BFD minimum interval. Range is 4-30000 milliseconds.
Step 7	**bfd multiplier** *multiplier* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)#bfd multiplier 7`	Sets the BFD multiplier. This is optional, the minimum is 3 and by default the multiplier will be 3 for all protocols
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Enabling BFD for OSPF on an Interface

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

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**router ospf** *process-name* **Example:**	Enters OSPF configuration mode, allowing you to configure the OSPF routing process. **Note**

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config)# router ospf 0`	To configure BFD for IS-IS, enter the corresponding configuration mode.
Step 3	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Configures an Open Shortest Path First (OSPF) area. Replace *area-id* with the OSPF area identifier.
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface TengigabitEthernet 0/3/0/1`	Enters interface configuration mode and specifies the interface name.
Step 5	**bfd fast-detect** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# bfd fast-detect`	Enables BFD to detect failures in the path between adjacent routers.
Step 6	**bfd minimum-interval** *milliseconds* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# bfd minimum-interval 6500`	Sets the BFD minimum interval. Range is 4-30000 milliseconds. This example sets the BFD minimum interval to 6500 milliseconds.
Step 7	**bfd multiplier** *multiplier* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# bfd multiplier 7`	Sets the BFD multiplier. This is optional, the minimum is 3 and by default the multiplier will be 3 for all protocols. This example sets the BFD multiplier to 7.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Enabling BFD on a Static Route

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

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**router static** **Example:** `RP/0/RP0/CPU0:router(config)# router static`	Enters static route configuration mode, allowing you to configure static routing.
Step 3	**address-family ipv4 unicast** *address nexthop* **Example:** `RP/0/RP0/CPU0:router(config-static)# address-family ipv4 unicast 10.2.2.0/24 10.6.0.2`	Enables BFD fast-detection on the specified IPv4 unicast destination address prefix and on the forwarding next-hop address.
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-static)# interface TengigabitEthernet 0/3/0/1`	Enters interface configuration mode and specifies the interface name.
Step 5	**bfd fast-detect** **Example:** `RP/0/RP0/CPU0:router(config-static-if)# bfd fast-detect`	Enables BFD to detect failures in the path between adjacent forwarding engines.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

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

Enabling BFD Sessions on Bundle Members

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

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**interface Bundle-Ether** *bundle-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1`	Enters interface configuration mode for the specified bundle ID.
Step 3	**bfd address-family ipv4 fast-detect** **Example:** `RP/0/RP0/CPU0:router(config-if)# bfd address-family ipv4 fast-detect`	Enables IPv4 BFD sessions on bundle member links.
Step 4	**bfd mode ietf** **Example:** `RP/0/RP0/CPU0:router(config-if)# bfd mode ietf`	Enables IETF mode for BFD over bundle for the specified bundle.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Specifying the BFD Destination Address on a Bundle

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

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**interface Bundle-Ether** *bundle-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1`	Enters interface configuration mode for the specified bundle ID.
Step 3	**bfd address-family ipv4 destination** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-if)# bfd address-family ipv4 destination 10.20.20.1`	Specifies the primary IPv4 address assigned to the bundle interface on a connected remote system, where *ip-address* is the 32-bit IP address in dotted-decimal format (A.B.C.D).
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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:

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**interface Bundle-Ether** *bundle-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1`	Enters interface configuration mode for the specified bundle ID.
Step 3	**bundle minimum-active bandwidth** *kbps* **Example:** `RP/0/RP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000`	Sets the minimum amount of bandwidth required before a bundle can be brought up or remain up. The range is from 1 through a number that varies depending on the platform and the bundle type.
Step 4	**bundle minimum-active links** *links* **Example:** `RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2`	Sets the number of active links required before a bundle can be brought up or remain up. The range is from 1 to 32. **Note** When BFD is started on a bundle that is already active, the BFD state of the bundle is declared when the BFD state of all the existing active members is known.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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

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

The BFD control packet interval is configured directly using the **bfd address-family ipv4 minimum-interval** command. The failure detection times are determined by a combination of the interval and multiplier values in these commands.

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

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**interface Bundle-Ether** *bundle-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1`	Enters interface configuration mode for the specified bundle ID.
Step 3	**bfd address-family ipv4 minimum-interval** *milliseconds* **Example:** `RP/0/RP0/CPU0:router(config-if)#bfd address-family ipv4 minimum-interval 2000` **Note** Specifies the minimum interval, in milliseconds, for asynchronous mode control packets on IPv4 BFD sessions on bundle member links. The range is from 4 to 30000.	
Step 4	**bfd address-family ipv4 multiplier** *multiplier* **Example:** `RP/0/RP0/CPU0:router(config-if)#bfd address-family ipv4 multiplier 30`	Specifies a number that is used as a multiplier with the minimum interval to determine BFD control packet failure detection times and transmission intervals for IPv4 BFD sessions on bundle member links. The range is from 2 to 50. The default is 3. **Note** Although the command allows you to configure a minimum of 2, the supported minimum is 3.

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

Configuring BFD over Bundle per Member Mode

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**bfd bundle per-member mode ietf** **Example:** `RP/0/RP0/CPU0:router(config)# bfd bundle per-member mode ietf`	Enables IETF mode for BFD over per-bundle member link.
Step 3	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configure BFD over Bundles IETF Mode Support on a Per Bundle Basis

To configure BFD over Bundles IETF mode support on a per bundle basis use these steps:

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**interface Bundle-Ether** *bundle-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1`	Enters interface configuration mode for the specified bundle ID.
Step 3	**bfd mode ietf** **Example:** `RP/0/RP0/CPU0:router(config-if)# bfd mode ietf`	Enables IETF mode for BFD over bundle for the specified bundle.
Step 4	**bfd address-family ipv4 fast-detect** **Example:** `RP/0/RP0/CPU0:router(config-if)# bfd address-family ipv4 fast-detect`	Enables IPv4 BFD sessions on the specified bundle.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 6	**show bundle bundle-ether** *bundle-id*	Displays the selected bundle mode.

BoB Configuration for IPv4 and IPv6

Table 24: Feature History

Feature Name	Release Information	Feature Description
BFD v6 - HW Offload and IPv6 BFD/BoB (Bundle over Bundle)	Release 7.3.1	The Bidirectional Forwarding detection (BFD) Hardware Offload feature enables the offload of a BFD session in an IPv6 network. With this feature, each bundle member link with IPv6 address runs its own BFD session This feature improves scale and reduces the overall network convergence time by sending rapid failure detection packets to the routing protocols for recalculating the routing table.

The Bidirectional Forwarding detection (BFD) Hardware Offload feature enables the offload of a BFD session to the network processing units of the line cards, in an IPv4 network. BFD hardware offload improves scale and reduces the overall network convergence time by sending rapid failure detection packets to the routing protocols for recalculating the routing table.

Restrictions

- BFD over Bundle (BOB) over IPv6 is not supported with dynamically configured link-local address. It must be statically configured.
- In Cisco IOS XR Release 26.2.2 and Cisco IOS XR Release 26.3.1, **show bfd session** commands that include the **hardware** keyword can display this warning:

  ```
  sysdb_list failed 0x40818800,
          'sysdb' detected the 'warning' condition 'An invalid argument was passed to a
  SysDB function
          (maybe a NULL pointer?)'
  ```

  The output can be incomplete when hardware-offloaded sessions require multiple result pages. This warning does not affect BFD sessions or packet forwarding.

  To avoid the warning, run **show bfd session** command without the **hardware** keyword. Check the `H/W` and `NPU` columns to verify hardware-offload status.

Confiugration Example

Configuration example for IPv4

```
/* Configure BFD over Bundle(BOB) for hardware offload. */
Router# config
Router(config)# interface Bundle-Ether 1
Router(config-if)# bfd mode ietf
Router(config-if)# bfd address-family ipv4 multiplier 3
Router (config-if)# bfd address-family ipv4 destination 10.20.20.1
Router (config-if)# bfd address-family ipv4 fast-detect
Router(config-if)# bfd address-family ipv4 minimum-interval 2000
Router(config-if)# ipv4 address 10.20.20.2/30
```

Configuration example for IPv6

```
/* Configure BFD over Bundle(BOB) for hardware offload. */
Router# config
Router(config)# interface Bundle-Ether 1
Router(config-if)# bfd mode ietf
Router(config-if)# bfd address-family ipv6 multiplier 3
Router (config-if)# bfd address-family ipv6 destination 10.20:20::1
Router (config-if)# bfd address-family ipv6 fast-detect
Router(config-if)# bfd address-family ipv6 minimum-interval 2000
Router(config-if)# ipv6 address 10:20:20::2/64
```

Configuration Verification

Configuration example for IPv4

Use the **show bfd ipv4 session** command to verify the BoB Configuration for IPv4:

```
Router#show bfd ipv4 session
Interface           Dest Addr           Local det time(int*mult)      State
                                    Echo             Async   H/W   NPU
------------------- --------------- ---------------- ---------------- ----------
Hu0/0/0/22           10.20.20.1      0s(0s*0)         6s(2s*3)        UP
                                                             Yes     0/0/CPU0
BE1                 10.20.20.1      n/a              n/a              UP
                                                             No      n/a
```

Configuration example for IPv6

Use the **show bfd ipv6 session** command to verify the BoB Configuration for IPv6:

```
Router#show bfd ipv6 session
Interface           Dest Addr
                                        Local det time(int*mult)      State
H/W                 NPU             Echo             Async
------------------- --------------- ---------------- ---------------- ----------
Hu0/0/0/1           10.20:20::1
Yes                 0/0/CPU0        0s(0s*0)         6s(2s*3)             UP
BE1                 10.20:20::1
No                  n/a             n/a              n/a              UP
```

BFD Hardware Offload Support for IPv6

Table 25: Feature History

Feature Name	Release Information	Feature Description
BFD v6 - HW Offload and IPv6 BFD/BoB (Bundle over Bundle)	Release 7.3.1	The Bidirectional Forwarding detection (BFD) Hardware Offload feature enables the offload of a BFD session in an IPv6 network. With this feature, each bundle member link with IPv6 address runs its own BFD session This feature improves scale and reduces the overall network convergence time by sending rapid failure detection packets to the routing protocols for recalculating the routing table.

The Bidirectional Forwarding detection (BFD) Hardware Offload feature enables the offload of a BFD session to the network processing units of the line cards, in an IPv6 network. BFD hardware offload feature improves scale and reduces the overall network convergence time by sending rapid failure detection packets to the routing protocols for recalculating the routing table.

Restrictions

- This feature is not supported over MPLS LDP interfaces.
- This feature is not supported over MPLS TE or RSVP tunnel.
- BFD Dampening is not supported for BFD over IPv6.
- BFD over Bundle (BOB) over IPv6 is not supported with dynamically configured link-local address. It must be statically configured.
- BFD multihop will flap if underlay paths that consist of multiple bundle VLANs flap.

Configuration Example

```
/* Configure BFD over Bundle(BOB) for hardware offload. */
Router# config
Router(config)# interface Bundle-Ether 1
Router(config-if)# bfd mode ietf
Router(config-if)# bfd address-family ipv6 multiplier 3
Router (config-if)# bfd address-family ipv6 destination 10.20:20::1
Router (config-if)# bfd address-family ipv6 fast-detect
Router(config-if)# bfd address-family ipv6 minimum-interval 2000
Router(config-if)# ipv6 address 10:20:20::2/64

/* To define the line card to host BLB and BFD multihop sessions. */
Router(config)# bfd
Router(config-bfd)# multipath include location 0/RP0/CPU0

/* Configure BFD with a static route. */
Router(config)# router static
Router(config-static)# address-family ipv6 unicast 1011:17e4::1/128 ab11:15d2::2 bfd
fast-detect minimum-interval 50 multiplier 3

/* Configure BFD with IS-IS. */
Router(config)# router isis 65444
Router(config-isis)# address-family ipv6 unicast
Router(config-isis)# exit
Router(config-isis)# interface gigabitEthernet 0/3/0/1
Router(config-isis-if)#  bfd minimum-interval 6500
Router(config-isis-if)# bfd multiplier 7
Router(config-isis-if)# bfd fast-detect ipv6
Router(config-isis-if)# address-family ipv6 unicast

/* Configure BFDv6 with OSPFv3. */
Router(config)# router ospfv3 main
Router(config-ospfv3)# area 0
Router(config-ospfv3-ar)#  interface gigabitEthernet 1/0/0/1
Router(config-ospfv3-ar-if)# bfd multiplier 7
Router(config-ospfv3-ar-if)# bfd fast-detect
Router(config-ospfv3-ar-if)# bfd minimum-interval 6500

/* Configuring BFD over BGP. */
Router(config)# router bgp 120
Router(config-bgp)# neighbor 2001:DB8:1::1
Router(config-bgp-nbr)# bfd fast-detect
```

```
Router(config-bgp-nbr)# bfd multiplier 7
Router(config-bgp-nbr)# bfd minimum-interval 6500
```

Verification

Use the **show bfd ipv6 session** command to verify the configuration:

```
Router# show bfd ipv6 session
Interface           Dest Addr
                                      Local det time(int*mult)      State
H/W                 NPU             Echo             Async
------------------- --------------- ---------------- ---------------- ----------
BE7.2               fe80::28a:96ff:fed6:9cdb
Yes                 0/RP0/CPU0        0s(0s*0)         900ms(300ms*3)   UP
BE7.4               fe80::28a:96ff:fed6:9cdb
Yes                 0/RP0/CPU0        0s(0s*0)         900ms(300ms*3)   UP
```

BFD over Bundle with IPv4 Unnumbered Interfaces

BFD over Bundle with IPv4 Unnumbered Interfaces feature enables BFD to run on IP unnumbered interfaces, which take the IP address from the loopback address. The same loopback address is used on multiple interfaces. This saves IP addresses space or range.

BFD creates a session on the unnumbered interface for which the BFD clients provide the source and destination IP address along with the interface index. BFD establishes the session on the Layer 3 unnumbered link to which the interface index corresponds. The source address is derived from the Loopback interface at the source. The destination node also uses IP unnumbered interface with loopback address and that is used as destination IP address.

BFD sends control packets to the unnumbered interfaces. These control packets are the regular IP BFD packets. Address Resolution Protocol (ARP) resolves the destination loopback IP address to the destination node's router MAC address.

Restriction

Only Asynchronous mode is supported.

Configure BFD over Bundle with IPv4 Unnumbered Interface

- Configure loopback address
- Add physical interface to bundle
- Configure BOB session on an unnumbered interface

Configure Loopback Address

```
Router(config)# interface loopback 1
Router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
```

Add Physical Interface to Bundle

```
Router(config)# interface HundredGigE0/0/1/0
Router(config-if)# bundle id 1 mode on
```

Configure a BFD over Bundle Session on an Unnumbered Interface

```
Router(config)# interface Bundle-Ether1
Router(config-if)# bfd address-family ipv4 destination 10.2.2.2
```

```
Router(config-if)# bfd address-family ipv4 fast-detect
Router(config-if)# ipv4 point-to-point
Router(config-if)# ipv4 unnumbered Loopback1
```

Running Configuration

```
interface Loopback1
ipv4 address 10.1.1.1 255.255.255.0
!
interface HundredGigE0/0/1/0
bundle id 1 mode on
!
interface Bundle-Ether1
bfd address-family ipv4 destination 10.2.2.2
bfd address-family ipv4 fast-detect
ipv4 point-to-point
ipv4 unnumbered Loopback1
```

Configuration Verification

```
show bfd session

Interface           Dest Addr           Local det time(int*mult)      State
                                          Echo              Async   H/W   NPU
------------------- --------------- ----------------- ---------------- ----------
Hu0/0/1/0           10.2.2.2      0s(0s*0)          450ms(150ms*3)   UP
                                                                     Yes   0/0/CPU0
BE1                 10.2.2.2      n/a               n/a               UP
                                                                     No    n/a
```

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 IPv4 Multihop BFD 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.

In case of BFD multipath sessions, the input and output interface may change based on the routing table updates. If the multipath session BFD packets must get preferential treatment, then a QoS policy must be configured on the entire path, including the possible input and output interfaces of the router.

Note The CLI **bfd multipath include location** *location* is a mandatory configuration to download BFD sessions on a given location.

BFD over BVI

Table 26: Feature History

Feature Name	Release Information	Feature Description
BFD on BVI	Release 7.3.1	BFD can be configured on Bridge group Virtual Interface (BVI). BVI is a virtual interface within the router that acts like a normal routed interface that does not support bridging but represents the bridge group for the bridged physical interfaces. BFD detects the Layer3 fault over the BVI much quicker and informs the same to routing protocols.

In order for a VLAN to span a router, the router must be capable of forwarding frames from one interface to another, while maintaining the VLAN header. If the router is configured for routing a Layer 3 (network layer) protocol, it will terminate the VLAN and MAC layers at the interface on which a frame arrives. The MAC layer header can be maintained if the router bridges the network layer protocol. However, even regular bridging terminates the VLAN header.

Using the Integrated Routing Bridging (IRB) feature, a router can be configured for routing and bridging the same network layer protocol, on the same interface. This allows the VLAN header to be maintained on a frame while it transits a router from one interface to another. IRB provides the ability to route between a bridged domain and a routed domain with the Bridge Group Virtual Interface (BVI). The BVI is a virtual interface within the router that acts like a normal routed interface that does not support bridging, but represents the comparable bridge group to routed interfaces within the router. The interface number of the BVI is the number of the bridge group that the virtual interface represents. This number is the link between the BVI and the bridge group.

Because the BVI represents a bridge group as a routed interface, it must be configured only with Layer 3 (L3) characteristics, such as network layer addresses. Similarly, the interfaces configured for bridging a protocol must not be configured with any L3 characteristics.

BFD over IRB is a multipath single-hop session. BFD over IRB is supported on IPv4 address, IPv6 global address, and IPv6 link-local address. The BFD over IRB is supported only in asynchronous mode and does not support echo mode.

IPv4 Multihop BFD

IPv4 Multihop BFD is a BFD session between two addresses that are several hops away. An example of this feature 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 IPv4 Multihop BFD are external and internal BGP. IPv4 Multihop BFD feature supports BFD on arbitrary paths, which can span multiple networks hops.

A Virtual Routing and Forwarding (VRF) instance is a logical separation of a router's routing table. VRF allows you to have multiple routing tables on a single router, each with its own set of routes.

The default VRF is the first VRF that is created on a router. It is the VRF that is used by default for all routing protocols and interfaces.

Non-default VRFs must be explicitly configured.

The IPv4 Multihop BFD feature provides subsecond forwarding failure detection for a destination more than one hop, and up to 255 hops, away. IPv4 Multihop BFD feature is supported on all currently supported media-type for BFD single hop.

You can set up a BFD multihop session between a unique source-destination address pair that is provided by the client. You can set up a session two endpoints that have IP connectivity.

Multihop BFD feature runs on both default and non-default VRF.

Configure IPv4 Multihop BFD

This section describes how you can configure IPv4 Multihop BFD feature.

```
Router# configure
Router(config)# bfd
Router(config)# multipath include location 0/7/CPU0
Router(config)# router bgp 100
Router(config-bgp)# neighbor 209.165.200.225
Router(config-bgp-nbr)# remote-as 2000
Router(config-bgp-nbr)# update-source loopback 1
Router(config-bgp-nbr)# bfd fast-detect
Router(config-bgp-nbr)# bfd multiplier 3
Router(config-bgp-nbr)# bfd minimum-interval 300
Router(config-bgp-nbr-af)# commit
```

Running Configuration

```
bfd
 multipath include location 0/7/CPU0
router bgp 100
 neighbor 209.165.200.225
  remote-as 2000
  update-source loopback 1
  bfd fast-detect
  bfd multiplier 3
  bfd minimum-interval 300
address-family ipv4 unicast
```

Configure Multihop BFD on IPv4 Non-default VRFs

Configure multihop BFD on IPv4 or IPv6 non-default VRFs:

- Configure BGP with the Autonomous System Number (ASN) on the router.

- Define a BGP neighbor with the specified IPv4 or IPv6 address.
- Associate the neighbor with a non-default VRF named "vrf1."
- Assign a route distinguisher value to create a routing and forwarding table for a VRF.
- Configure the redistribution of connected routes.
- Establish and configure an eBGP session with the specified IPv4 or IPv6 neighbor.
- Configure the remote ASN.
- Enable BFD for fast link failure detection.
- Set the BFD detection time parameters.
- Configure eBGP sessions.
- Specify the primary IP address from a particular interface as the local address when forming an eBGP session with a neighbor.
- Apply a route-policy for both inbound and outbound traffic.

Configure the following steps to configure Multihop BFD on IPv4 nondefault VRF:

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 209.165.200.225
Router(config-bgp-nbr)#vrf vrf1
Router(config-bgp-nbr-vrf)# exit
Router(config-bgp-nbr)# rd auto
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)#redistribute connected
Router(config-bgp-nbr-af)# exit
Router(config-bgp)# neighbor 209.165.200.225
Router(config-bgp-nbr)# remote-as 2000
Router(config-bgp-nbr)# bfd fast-detect
Router(config-bgp-nbr)# bfd multiplier 3
Router(config-bgp-nbr)# bfd minimum-interval 50
Router(config-bgp-nbr)# ebgp-multihop 255
Router(config-bgp-nbr)# update-source loopback 1
/* You can configure any interface here, including loopback or bvi */
Router(config-bgp-nbr)#address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# commit
```

Running Configuration

```
router bgp 100
  neighbor 209.165.200.225
   vrf vrf1
    exit
     rd auto
      address-family ipv4 unicast
       redistribute connected
        exit
         neighbor 209.165.200.225
         remote-as 2000
         bfd fast-detect
         bfd multiplier 3
         bfd minimum-interval 50
```

```
ebgp-multihop 255
update-source loopback 1
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
```

Verification

```
Router# show bfd session source 209.165.200.225
Thu Mar 10 10:13:43.305 IST
Src Addr         Dest Addr    VRF Name         H/W NPU
                               Local det time(int*mult)   State
                              Echo    Async
--------------   ---------   ------   ------ --------- ---------
209.165.200.225  192.0.2.254   vrf_1     Yes   0/0/CPU0
                              n/a  150ms(50ms*3)          UP
Router# show cef vrf vrf_1 209.165.200.225 location 0/0/CPU0
Thu Mar 10 10:24:13.372 IST
209.165.200.0/24, version 40, internal 0x5000001 0x30 (ptr 0x8ae26458) [1], 0x0 (0x0), 0xa08
 (0x8dc144a8)
 Updated Mar  9 15:09:43.398
 Prefix Len 24, traffic index 0, precedence n/a, priority 3
LDI Update time Mar  9 14:59:28.284
   via 1.1.1.1/32, 605 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x8dd35988 0x0]
    recursion-via-/32
    next hop VRF - 'default', table - 0xe0000000
    next hop 10.1.1.1/32 via 24015/0/21
    next hop 192.0.2.255/32 Te0/0/0/3.1  labels imposed {ImplNull 24162}
```

Multihop BFD over BVI

Table 27: Feature History Table

Feature Name	Release Information	Feature Description
Multihop BFD over Bridge Group Virtual Interface (BVI)	Release 7.4.1	The multihop BFD over Bridge Group Virtual Interface (BVI) feature introduces support for multihop BFD over (BVI). You can set up a multihop BFD session between two endpoints that have IP connectivity. This session is between a unique source-destination address pair that the client provides. This feature allows you to extend BFD on arbitrary paths. These arbitrary paths can span multiple network hops, hence detecting link failures.

Multihop BFD over BVI feature allows you to configure both routing and bridging on the same interface using Integrated Routing Bridging (IRB). IRB enables you to route between a bridged domain and a routed domain with the Bridge Group Virtual Interface (BVI).

The BVI is a virtual interface within the router that acts like a normal, routed interface that does not support bridging, but represents the comparable bridge group to routed interfaces within the router.

Restrictions

- The minimum Multihop BFD timer for the BVI interface is 50 msec.

- The **multihop ttl-drop-threshold** command is not supported.
- The Multihop BFD over BVI or IRB functionality is supported only in asynchronous mode and does not support echo mode.
- The Multihop BFD over BVI feature is not supported over MPLS and SR core.

Supported Functionality

- This feature is supported in both IPv4 and IPv6.
- BFD Multihop over BVI feature supports on client BGP.
- BFD Multihop supports only over IP core.
- BFD Multihop supports on all currently supported media-type for BFD single-hop.

Configuration

```
/* Configure a BVI interface and assign an IP address */
Router(config)# interface BVI1
Router(config-if)# host-routing
Router(config-if)# mtu 8986
Router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
Router(config-if)# ipv6 address 10:1:1::1/120

/* Configure the Layer 2 AC interface */
Router(config-if)# interface TenGigE0/5/0/6/0.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag pop 1 symmetric

/* Configure L2VPN Bridge Domain */
Router(config-subif)# l2vpn
Router(config-subif)# bridge group 1
Router(config-subif)# bridge-domain 1
Router(config-l2vpn-bg-bd)# interface TenGigE0/5/0/6/0.1
Router(config-l2vpn-bg-bd)# routed interface BVI1
```

Running Configuration

```
interface BVI1
 host-routing
 mtu 8986
 ipv4 address 10.1.1.1 255.255.255.0
 ipv6 address 10:1:1::1/120
!
interface TenGigE0/5/0/6/0.1 l2transport
 encapsulation dot1q 1
 rewrite ingress tag pop 1 symmetric
!
l2vpn
 bridge group 1
  bridge-domain 1
   interface TenGigE0/5/0/6/0.1
   !
   routed interface BVI1
   !
```

Repeat the configuration on the peer router.

```
/* Configure BGP as the routing protocol  */
Router(config)# router bgp 1
Router(config-bgp)# neighbor 2.2.1.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# bfd fast-detect
Router(config-bgp-nbr)# bfd minimum-interval 300
Router(config-bgp-nbr)# update-source Loopback1
Router(config-bgp-nbr)# address-family ipv4 unicast

/* Configure reachability to the BGP neighbour IP either via static or IGP*/
Router(config-bgp-nbr-af)# router static
Router(config-static)# address-family ipv4 unicast
Router(config-static-afi)# 2.2.1.1/32 10.1.1.2

/* Configure the line cards to allow hosting of Multipath BFD sessions. */
Router(config-static-afi)# bfd
Router(config-bfd)#
 multipath include location 0/RP0/CPU0


router bgp 1
neighbor 2.2.1.1
  remote-as 1
  bfd fast-detect
  bfd minimum-interval 300
  update-source Loopback1
  address-family ipv4 unicast
  !
router static
 address-family ipv4 unicast
  2.2.1.1/32 10.1.1.2
 !
bfd
 multipath include location 0/RP0/CPU0!
```

Note To avoid the unsupported three-level recursion on BVI interfaces on the first and second generation of line cards, you must not configure the BVI interface as the next-hop in the static route configuration.

Verification

```
Router# show bfd session destination 2.2.1.1
Fri May 28 14:35:52.566 IST

Src Addr            Dest Addr       VRF Name                        H/W NPU
                                        Local det time(int*mult)        State
                                    Echo             Async
------------------- --------------- ---------------- ---------------- ----------
1.1.1.1             2.2.1.1         default                          Yes 0/RP0/CPU0
                                    n/a              900ms(300ms*3)   UP
```

Seamless Bidirectional Forwarding Detection

Table 28: Feature History Table

Feature Name	Release Information	Feature Description
Seamless Bidirectional Forwarding Detection	Release 24.2.11	This feature now extends support on the Cisco NCS 540 Series routers running on Cisco IOS XR7.
Seamless Bidirectional Forwarding Detection	Release 24.2.1	Introduced in this release on the following Cisco NCS 540 router variants running on Cisco IOS XR: • N540-ACC-SYS • N540X-ACC-SYS • N540-24Z8Q2C-SYS This feature introduces support for NCS 5500 routers as a Seamless BFD (S-BFD) reflector. Seamless BFD simplifies the negotiation and state establishment aspects of BFD by predetermining session discriminators and maintaining session state only at the headend. This approach ensures quicker connectivity tests and reduces complexity in session establishment. Previously, support for Seamless BFD reflector was not available. The feature introduces these changes: **CLI:** This feature introduces the **sbfd** command.

Advantages of SBFD over BFD

Seamless Bidirectional Forwarding Detection (S-BFD), is a simplified mechanism for using BFD with a large proportion of negotiation aspects eliminated, thus providing benefits such as quick provisioning, as well as improved control and flexibility for network nodes initiating path monitoring.

Components of S-BFD

S-BFD includes the following components:

- S-BFD discriminator
- Reflector BFD session
- S-BFD initiator

Each network node allocates one or more S-BFD discriminators for local entities and creates a reflector BFD session. The S-BFD initiator sends S-BFD control packets with the corresponding discriminator value. The reflector BFD session listens to incoming S-BFD control packets addressed to local entities and generates response S-BFD control packets.

Key differences between BFD and S-BFD

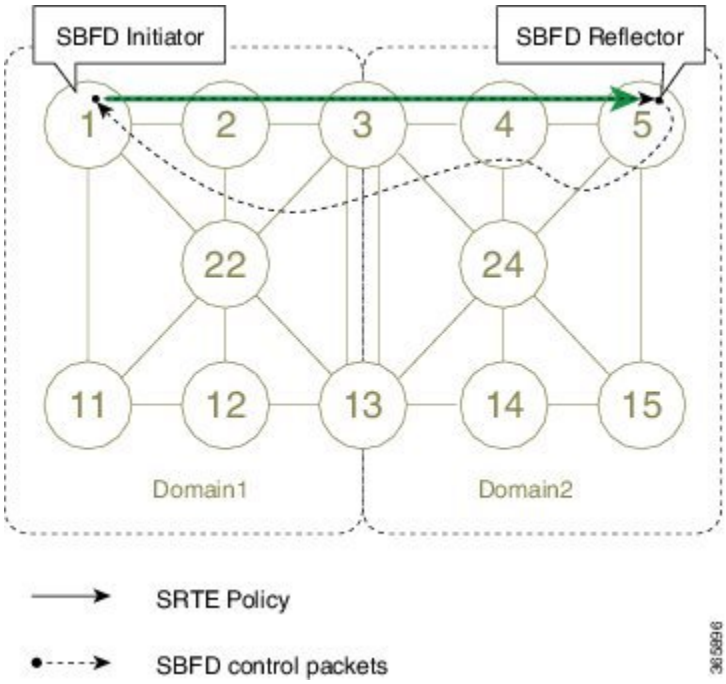
In BFD, each end of the connection maintains a BFD state and transmits packets periodically over a forwarding path. S-BFD is unidirectional, resulting in faster session activation than BFD. The BFD state and client context is maintained on the head-end (initiator) only. The tail-end (reflector) validates the BFD packet and responds, so there is no need to maintain the BFD state on the tail-end.

Initiator and Reflector Components of S-BFD

S-BFD runs in an asymmetric behavior, using initiators and reflectors.

The following figure represents the roles of the S-BFD initiator and reflector.

Figure 7: S-BFD Initiator and Reflector



The initiator is an S-BFD session on a network node that performs a continuity test to a remote entity by sending S-BFD packets. The initiator injects the S-BFD packets into the segment-routing traffic-engineering (SRTE) policy. The initiator triggers the S-BFD session and maintains the BFD state and client context. For more information about configuring SRTE policies, see the *Configure SR-TE Policies* chapter in the *Segment Routing Configuration Guide*.

The S-BFD reflector is an S-BFD session on a network node that listens for incoming S-BFD control packets to local entities and generates response S-BFD control packets. The reflector is stateless and only reflects the S-BFD packets back to the initiator.

Role of Discriminators in S-BFD Control Packet

The BFD control packet carries 32-bit discriminators (local and remote) to demultiplex BFD sessions. S-BFD requires globally unique S-BFD discriminators that are known by the initiator.

The S-BFD control packets contain the discriminator of the initiator, which is created dynamically by the initiator, and the discriminator of the reflector, which is configured as a local discriminator on the reflector.

Usage Guidelines and Limitations for S-BFD

The following usage guidelines and limitations apply:

- The NCS 5500 routers do not support initiator mode.
- The feature support is only for a global VRF and IPv4 addresses.
- The supported Packets Per Second (PPS) limit is up to 3000. Also, consider the jitter used by the initiator for accurate performance assessment.
- The network administrator configures reflector node discriminators at the initiator, allowing the initiator to know the globally unique discriminators of the reflector before the session starts.
- The S-BFD over SR policy is not supported on NCS 540 routers.

Configure the S-BFD Reflector

This section includes steps to configure the S-BFD reflector.

Before you begin

- Each reflector should have at least one globally unique discriminator, to ensure the S-BFD packet arrives on the intended reflector.
- An S-BFD reflector only accepts BFD control packets where "Your Discriminator" is the reflector discriminator.

Procedure

Step 1 Configure the local discriminators on the reflector using the **sbfd local-discriminator** {*ipv4-address* | *32-bit-value* | **dynamic** | **interface** *interface*} command.

You can configure a local discriminator in one of the following ways. For more information about configuring a local discriminator, see the *local-discriminator* command in the *Segment Routing Command Reference for Cisco 5500 Series Routers*.

- Configure an IPv4 address as the local discriminator.

```
Router(config)#sbfd
Router(config-sbfd)#local-discriminator 192.0.2.1
```

- Configure a unique 32-bit value as the local discriminator.

```
Router(config)#sbfd
Router(config-sbfd)#local-discriminator 987654321
```

- Configure an IPv4 address of the interface as the local discriminator.

```
Router(config)#sbfd
Router(config-sbfd)#local-discriminator interface Loopback0
```

- Configure a randomly generated value as the local discriminator.

```
Router(config)#sbfd
Router(config-sbfd)#local-discriminator dynamic
```

Step 2 Verify the configuration using the **show running-config** command.

Example:

```
 local-discriminator 10.1.1.5
 local-discriminator 987654321
 local-discriminator dynamic
 local-discriminator interface Loopback0
!
```

Step 3 Verify the configured BFD local discriminators using the **show bfd target-identifier** command.

Example:

```
Router#show bfd target-identifier local

Local Target Identifier Table
-----------------------------
Discr       Discr Src   VRF       Status   Flags
                        Name
-----       ---------   -------   -------- --------
16843013    Local       default   enable   -----ia-
987654321   Local       default   enable   ----v---
2147483649  Local       default   enable   -------d

Legend: TID - Target Identifier
        a   - IP Address mode
        d   - Dynamic mode
        i   - Interface mode
        v   - Explicit Value mode
```

Step 4 Verify the S-BFD reflector configuration using the **show bfd reflector** command.

Example:

```
Router#show bfd reflector info detail location 0/0/CPU0

Local Discr     : 2147483649
Remote Discr    : 65576
Source Address  : 1.1.1.1
Last DOWN received Time : (NA)
Last Rx packets timestamps before DOWN
  [NA                 ] [NA                 ] [NA                 ] [NA                 ]
```

```
  [NA                 ] [NA                 ] [NA                 ] [NA                 ]
  [NA                 ] [NA                 ]
Last Tx packets timestamps before DOWN
  [NA                 ] [NA                 ] [NA                 ] [NA                 ]
  [NA                 ] [NA                 ] [NA                 ] [NA                 ]
  [NA                 ] [NA                 ]
Last UP sent Time : (Jun  7 14:59:34.763)
Last recent Rx packets timestamps:
  [Jun 7 15:00:18.653 ] [Jun 7 15:00:18.751 ] [Jun 7 15:00:18.837 ] [Jun 7 15:00:18.927 ]
  [Jun 7 15:00:18.085 ] [Jun 7 15:00:18.185 ] [Jun 7 15:00:18.274 ] [Jun 7 15:00:18.372 ]
  [Jun 7 15:00:18.464 ] [Jun 7 15:00:18.562 ]
Last recent Tx packets timestamps:
  [Jun 7 15:00:18.653 ] [Jun 7 15:00:18.751 ] [Jun 7 15:00:18.837 ] [Jun 7 15:00:18.927 ]
  [Jun 7 15:00:18.085 ] [Jun 7 15:00:18.185 ] [Jun 7 15:00:18.274 ] [Jun 7 15:00:18.372 ]
  [Jun 7 15:00:18.464 ] [Jun 7 15:00:18.563 ]
```

Coexistence Between BFD over Bundle and BFD over Logical Bundle

The coexistence between BFD over Bundle (BoB) and BFD over Logical Bundle (BLB) feature allows you to monitor either physical bundle member for BOB, or logical interface for BLB, or both. This feature enables BFD to converge fast.

Difference between BoB and BLB

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

BFD over Logical Bundle (BLB) (RFC 5880) treats a bundle interface with all its members as a single interface. BLB is a multipath (MP) single-hop session. If BLB is configured on a bundle there is only one single BFD session that is active. This implies that only one bundle member is being monitored by BFD at any given time. The client is one of the routing protocols. When BFD detects a failure, the client brings down the routing session.

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

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

Coexistence modes for BoB and BLB

Logical coexistence mode:

When BoB and BLB coexistence is configured in logical mode, only BLB sessions using the main bundle interface's IPv4 address are inherited from the BoB session. BLB sessions configured on bundle sub-interfaces, each with a unique IPv4 address, will establish independent sessions. BLB sessions on loopback interfaces associated with the bundle also operate separately from the main bundle interface.

Inherit Coexistence Mode

In inherit mode, the system does not initiate a standalone BLB session on either the main interface or the sub-interfaces if BoB is not configured on the bundle. In this situation, the BLB session stays in a virtual *Down* state until a corresponding BoB session becomes available. In inherit mode, the BLB session always depends on the inherited BoB session and does not establish its own BFD session with real packets unless BoB is enabled on the bundle. Therefore, without BoB configured on the bundle interfaces, no BLB session will operate in inherit mode.

Functionality Comparison

Logical mode provides greater flexibility by allowing BLB sessions to be established independently on sub-interfaces or loopback interfaces. Inherit mode restricts BLB to inherit only from existing BoB sessions and does not allow independent BLB session establishment.

Note

- Use logical mode when independent BLB sessions per sub-interface or loopback are required.
- Use inherit mode when BLB session activation must be strictly dependent on the presence of a BoB session on the corresponding interface.

Configuration Example

Configure one or more linecards to allow hosting of MP BFD sessions. If no linecards are included, linecards groups are not formed, and consequently no BFD MP sessions are created. For default settings of group size and number, you must add at least two lines with the **bfd multiple-paths include location** *node-id* command and valid line cards to the configuration for the algorithm to start forming groups and BFD MP sessions to be established.

```
Router(config)# bfd multipath include location 0/RP0/CPU0
Router(config)# bfd multipath include location 0/1/CPU0

/* Configure inherited coexistence mode */
Router(config)# bfd
Router(config-bfd)# bundle coexistence bob-blb inherit

/* Configure logical coexistence mode */
Router(config)# bfd
Router(config-bfd)# bundle coexistence bob-blb logical
```

Running Configuration

Running confiiguration for inherited coexistence:

```
bfd
bundle coexistence bob-blb inherit
```

Running confiiguration for logical mode:

```
bfd
bundle coexistence bob-blb logical
```

Verification

Verify BOB and BLB coexistence inherited mode.

```
Router# show bfd session
Mon May 31 02:55:44.584 UTC
Interface           Dest Addr           Local det time(int*mult)      State
                                      Echo             Async   H/W   NPU
```

```
------------------- --------------- ---------------- ---------------- ----------
Te0/0/0/7           33.33.33.2      0s(0s*0)         450ms(150ms*3)   UP
                                                             Yes   0/RP0/CPU0
BE123               33.33.33.2      n/a              n/a              UP
                                                             No    n/a
BE123.1             34.34.34.2      n/a              n/a              UP
                                                             No    n/a

Router# show bfd session interface bundle-ether 123 detail
Fri May 28 13:49:35.268 UTC
I/f: Bundle-Ether123, Location: 0/RP0/CPU0
Dest: 33.33.33.2
Src: 33.33.33.1
 State: UP for 0d:0h:29m:50s, number of times UP: 1
 Session type: PR/V4/SH/BI/IB
Session owner information:
                          Desired               Adjusted
  Client                Interval   Multiplier Interval   Multiplier
  -------------------- ---------------------- ---------------------
  bundlemgr_distrib     150 ms     3          150 ms     3
Session association information:
  Interface             Dest Addr / Type
  -------------------- ----------------------------------
  Te0/0/0/7             33.33.33.2
                        BFD_SESSION_SUBTYPE_RTR_BUNDLE_MEMBER
  BE123.1               34.34.34.2
                        BFD_SESSION_SUBTYPE_STATE_INHERIT

Router# show bfd session interface bundle-ether 123.1 detail
Fri May 28 13:49:59.316 UTC
I/f: Bundle-Ether123.1, Location: 0/RP0/CPU0
Dest: 34.34.34.2
Src: 34.34.34.1
 State: UP for 0d:0h:12m:54s, number of times UP: 1
 Session type: PR/V4/SH/IH
Session owner information:
                          Desired               Adjusted
  Client                Interval   Multiplier Interval   Multiplier
  -------------------- ---------------------- ---------------------
  ipv4_static           100 ms     3          100 ms     3
Session association information:
  Interface             Dest Addr / Type
  -------------------- ----------------------------------
  BE123                 33.33.33.2
                        BFD_SESSION_SUBTYPE_RTR_BUNDLE_INTERFACE

Router# show bfd session interface tenGigE 0/0/0/7 detail
Mon May 31 03:00:04.635 UTC
I/f: TenGigE0/0/0/7, Location: 0/0/CPU0
Dest: 33.33.33.2
Src: 33.33.33.1
 State: UP for 2d:13h:40m:19s, number of times UP: 1
 Session type: PR/V4/SH/BM/IB
Received parameters:
 Version: 1, desired tx interval: 150 ms, required rx interval: 150 ms
 Required echo rx interval: 0 ms, multiplier: 3, diag: None
 My discr: 2147493276, your discr: 2147492184, state UP, D/F/P/C/A: 0/0/0/1/0
Transmitted parameters:
 Version: 1, desired tx interval: 150 ms, required rx interval: 150 ms
 Required echo rx interval: 0 ms, multiplier: 3, diag: None
 My discr: 2147492184, your discr: 2147493276, state UP, D/F/P/C/A: 0/0/0/1/0
Timer Values:
 Local negotiated async tx interval: 150 ms
 Remote negotiated async tx interval: 150 ms
 Desired echo tx interval: 0 s, local negotiated echo tx interval: 0 ms
```

```
 Echo detection time: 0 ms(0 ms*3), async detection time: 450 ms(150 ms*3)
Local Stats:
 Intervals between async packets:
   Tx: Number of intervals=4, min=5 ms, max=15 s, avg=6927 ms
       Last packet transmitted 222007 s ago
   Rx: Number of intervals=15, min=3 ms, max=1700 ms, avg=1133 ms
       Last packet received 222018 s ago
 Intervals between echo packets:
   Tx: Number of intervals=0, min=0 s, max=0 s, avg=0 s
       Last packet transmitted 0 s ago
   Rx: Number of intervals=0, min=0 s, max=0 s, avg=0 s
       Last packet received 0 s ago
 Latency of echo packets (time between tx and rx):
   Number of packets: 0, min=0 ms, max=0 ms, avg=0 ms
Session owner information:
                         Desired               Adjusted
  Client               Interval   Multiplier Interval   Multiplier
  -------------------- ---------------------- ----------------------
  bundlemgr_distrib    150 ms     3          150 ms     3
Session association information:
  Interface            Dest Addr / Type
  -------------------- -----------------------------------
  BE123                33.33.33.2
                       BFD_SESSION_SUBTYPE_RTR_BUNDLE_INTERFACE

H/W Offload Info:
 H/W Offload capability : Y, Hosted NPU     : 0/RP0/CPU0
 Async Offloaded        : Y, Echo Offloaded : N
 Async rx/tx            : 122/51

Platform Info:
NPU ID: 0
Async RTC ID        : 1          Echo RTC ID        : 0
Async Feature Mask  : 0x0        Echo Feature Mask  : 0x0
Async Session ID    : 0x2158     Echo Session ID    : 0x0
Async Tx Key        : 0x80002158  Echo Tx Key        : 0x0
Async Tx Stats addr : 0x0   Echo Tx Stats addr : 0x0
Async Rx Stats addr : 0x0   Echo Rx Stats addr : 0x0
```

Verify BOB and BLB coexistence logical mode.

```
show bfd session
Mon May 31 02:54:07.442 UTC
Interface           Dest Addr           Local det time(int*mult)      State
                                        Echo             Async   H/W   NPU
------------------- --------------- ---------------- ---------------- ----------
Te0/0/0/7           33.33.33.2      0s(0s*0)         450ms(150ms*3)   UP
                                                             Yes   0/0/CPU0
BE123.1             34.34.34.2      0s(0s*0)         300ms(100ms*3)   UP
                                                             Yes   0/0/CPU0
BE123               33.33.33.2      n/a              n/a              UP
                                                             No    n/a

Router# show bfd session interface bundle-ether 123 detail
Fri May 28 14:04:41.331 UTC
I/f: Bundle-Ether123, Location: 0/RP0/CPU0
Dest: 33.33.33.2
Src: 33.33.33.1
 State: UP for 0d:0h:44m:56s, number of times UP: 1
 Session type: PR/V4/SH/BI/IB
Session owner information:
                         Desired               Adjusted
  Client               Interval   Multiplier Interval   Multiplier
  -------------------- ---------------------- ----------------------
  bundlemgr_distrib    150 ms     3          150 ms     3
```

```
Session association information:
  Interface            Dest Addr / Type
  -------------------- ------------------------------------
  Te0/0/0/7            33.33.33.2
                       BFD_SESSION_SUBTYPE_RTR_BUNDLE_MEMBER

Router# show bfd session interface tenGigE 0/0/0/7 detail
Mon May 31 03:04:25.714 UTC
I/f: TenGigE0/0/0/7, Location: 0/RP0/CPU0Dest: 33.33.33.2
Src: 33.33.33.1
 State: UP for 2d:13h:44m:40s, number of times UP: 1
 Session type: PR/V4/SH/BM/IB
Received parameters:
 Version: 1, desired tx interval: 150 ms, required rx interval: 150 ms
 Required echo rx interval: 0 ms, multiplier: 3, diag: None
 My discr: 2147493276, your discr: 2147492184, state UP, D/F/P/C/A: 0/0/0/1/0
Transmitted parameters:
 Version: 1, desired tx interval: 150 ms, required rx interval: 150 ms
 Required echo rx interval: 0 ms, multiplier: 3, diag: None
 My discr: 2147492184, your discr: 2147493276, state UP, D/F/P/C/A: 0/0/0/1/0
Timer Values:
 Local negotiated async tx interval: 150 ms
 Remote negotiated async tx interval: 150 ms
 Desired echo tx interval: 0 s, local negotiated echo tx interval: 0 ms
 Echo detection time: 0 ms(0 ms*3), async detection time: 450 ms(150 ms*3)
Local Stats:
 Intervals between async packets:
   Tx: Number of intervals=4, min=5 ms, max=15 s, avg=6927 ms
       Last packet transmitted 222268 s ago
   Rx: Number of intervals=15, min=3 ms, max=1700 ms, avg=1133 ms
       Last packet received 222279 s ago
 Intervals between echo packets:
   Tx: Number of intervals=0, min=0 s, max=0 s, avg=0 s
       Last packet transmitted 0 s ago
   Rx: Number of intervals=0, min=0 s, max=0 s, avg=0 s
       Last packet received 0 s ago
 Latency of echo packets (time between tx and rx):
   Number of packets: 0, min=0 ms, max=0 ms, avg=0 ms
Session owner information:
                         Desired               Adjusted
  Client               Interval   Multiplier Interval   Multiplier
  -------------------- ---------------------- ---------------------
  bundlemgr_distrib    150 ms     3          150 ms     3
Session association information:
  Interface            Dest Addr / Type
  -------------------- ------------------------------------
  BE123                33.33.33.2
                       BFD_SESSION_SUBTYPE_RTR_BUNDLE_INTERFACE

H/W Offload Info:
 H/W Offload capability : Y, Hosted NPU     : 0/RP0/CPU0
 Async Offloaded        : Y, Echo Offloaded : N
 Async rx/tx            : 122/51

Platform Info:
NPU ID: 0
Async RTC ID        : 1          Echo RTC ID        : 0
Async Feature Mask  : 0x0        Echo Feature Mask  : 0x0
Async Session ID    : 0x2158     Echo Session ID    : 0x0
Async Tx Key        : 0x80002158  Echo Tx Key        : 0x0
Async Tx Stats addr : 0x0   Echo Tx Stats addr : 0x0
Async Rx Stats addr : 0x0   Echo Rx Stats addr : 0x0

Router# show bfd session interface bundle-ether 123.1 detail
Fri May 28 14:04:46.893 UTC
```

```
I/f: Bundle-Ether123.1, Location: 0/0/CPU0
Dest: 34.34.34.2
Src: 34.34.34.1
 State: UP for 0d:0h:5m:18s, number of times UP: 1
 Session type: SW/V4/SH/BL
Received parameters:
 Version: 1, desired tx interval: 100 ms, required rx interval: 100 ms
 Required echo rx interval: 0 ms, multiplier: 3, diag: None
 My discr: 984, your discr: 124, state UP, D/F/P/C/A: 0/0/0/1/0
Transmitted parameters:
 Version: 1, desired tx interval: 100 ms, required rx interval: 100 ms
 Required echo rx interval: 0 ms, multiplier: 3, diag: None
 My discr: 124, your discr: 984, state UP, D/F/P/C/A: 0/1/0/1/0
Timer Values:
 Local negotiated async tx interval: 100 ms
 Remote negotiated async tx interval: 100 ms
 Desired echo tx interval: 0 s, local negotiated echo tx interval: 0 ms
 Echo detection time: 0 ms(0 ms*3), async detection time: 300 ms(100 ms*3)
Label:
 Internal label: 24000/0x5dc0
Local Stats:
 Intervals between async packets:
   Tx: Number of intervals=3, min=103 ms, max=19 s, avg=7023 ms
       Last packet transmitted 318 s ago
   Rx: Number of intervals=15, min=1 ms, max=1704 ms, avg=1315 ms
       Last packet received 318 s ago
 Intervals between echo packets:
   Tx: Number of intervals=0, min=0 s, max=0 s, avg=0 s
       Last packet transmitted 0 s ago
   Rx: Number of intervals=0, min=0 s, max=0 s, avg=0 s
       Last packet received 0 s ago
 Latency of echo packets (time between tx and rx):
   Number of packets: 0, min=0 ms, max=0 ms, avg=0 ms
MP download state: BFD_MP_DOWNLOAD_ACK
State change time: May 28 13:59:07.124
Session owner information:
                          Desired               Adjusted
  Client               Interval   Multiplier Interval   Multiplier
  -------------------- ---------- ---------- ---------- ----------
  ipv4_static          100 ms     3          100 ms     3

H/W Offload Info:
 H/W Offload capability : Y, Hosted NPU     : 0/0/CPU0
 Async Offloaded        : Y, Echo Offloaded : N
 Async rx/tx            : 16/4

Platform Info:
NPU ID: 0
Async RTC ID        : 1          Echo RTC ID        : 0
Async Feature Mask  : 0x0        Echo Feature Mask  : 0x0
Async Session ID    : 0x7c       Echo Session ID    : 0x0
Async Tx Key        : 0x7c  Echo Tx Key        : 0x0
Async Tx Stats addr : 0x0   Echo Tx Stats addr : 0x0
Async Rx Stats addr : 0x0   Echo Rx Stats addr : 0x0
```

BFD CPU Offload Support for IPv6

Table 29: Feature History Table

Feature Name	Release Information	Feature Description
BFD CPU offload support for IPv6	Release 26.1.1	Introduced in this release on: NCS 5500 fixed port routers This functionality is now supported on all Cisco NCS 5500 router variants.
BFD CPU offload support for IPv6	Release 24.4.1	You can now enable CPU offloading for IPv6 BFD sessions, allowing the CPU to handle packet transmission and reception directly. This feature provides you the flexibility to choose between hardware-offloaded and CPU-offloaded IPv6 BFD sessions based on your requirements. This feature introduces these changes: **CLI:** • **hw-module profile bfd offload disable-v6**

The BFD CPU offload support for IPv6 feature enables the offload of a BFD session to the CPU, in an IPv6 network.

You can enable CPU offload for IPv6 BFD sessions by using the command **hw-module profile bfd offload disable-v6** and then restarting the router. When CPU offload functionality is enabled, the CPU directly handles BFD sessions, managing packet transmission and reception without offloading the sessions to hardware. If you do not enable CPU offload, BFD sessions are offloaded to the hardware by default.

Benefits of BFD CPU Offload Support for IPv6

The BFD CPU Offload Support for IPv6 feature provides the following benefit:

- **Flexibility**: Supports both hardware and CPU offloaded sessions, providing you with the flexibility to choose between hardware-offloaded and CPU-offloaded IPv6 BFD sessions based on your requirements.

Note IPv6 BFD sessions can be either hardware-offloaded or CPU-offloaded, but both types cannot exist simultaneously.

Limitations for BFD CPU offload support for IPv6

Review these limitations before you use BFD CPU offload support for IPv6.

- In Release 24.4.1, the feature supported only select variants of NCS 5500 fixed port routers and NCS 540 routers, where IPv6 BFD sessions were hosted on the ARM processor. From Release 26.1.1, Cisco extends support to all NCS 5500 fixed port routers, as well as the NCS 540 and NCS 560 routers.

- In Release 24.4.1, only BFD over physical or VLAN interfaces was supported in CPU mode. From Release 26.1.1, BFD over Bundle (BoB), BFD over logical bundle (BLB), Bridge-Group Virtual Interface (BVI), and multihop (MH) sessions are supported.
- Scale limits must be managed to ensure that the rate does not exceed 640 packets per second (PPS). You can check the actual PPS using the **show bfd summary** command.
- For single-path sessions, the minimum-interval value must be greater than or equal to 100 milliseconds.
- For multi-path sessions, including BLB, BVI, and MH, the minimum-interval value must be greater than or equal to 300 milliseconds.
- The maximum supported scale is 64 IPv6 BFD sessions.
- The BFD agent process is responsible for handling packets. If it crashes or restarts, it can cause BFD sessions to flap.
- When you configure **hw-module profile offload** to prioritize route download or performance-monitoring hardware offload along with Bsync, the router does not support BFD CPU offload sessions.

Configure BFD CPU Offload Support for IPv6

Follow these steps to enable CPU offload for BFD IPv6 sessions:

Procedure

Step 1 Enable CPU Offload.

Example:

```
Router# configure
Router(config)# hw-module profile bfd offload disable-v6
```

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

Step 2 Verify if CPU offload is enabled by using the **show bfd ipv6 session** command.

Example:

```
Router# show bfd ipv6 session
Interface           Dest Addr
                                        Local det time(int*mult)      State
H/W                 NPU             Echo              Async
------------------- --------------- ----------------- ----------------- ----------
Te0/0/0/0.501       2001:DB8::1:2
No                  n/a             0s(0s*0)          300ms(100ms*3)    UP
Te0/0/0/0.502       2001:DB8::2:2
No                  n/a             0s(0s*0)          300ms(100ms*3)    UP
Te0/0/0/0.503       2001:DB8::3:2
No                  n/a             0s(0s*0)          300ms(100ms*3)    UP
Te0/0/0/0.504       2001:DB8::4:2
No                  n/a             0s(0s*0)          300ms(100ms*3)    UP
Te0/0/0/0.505       2001:DB8::5:2
No                  n/a             0s(0s*0)          300ms(100ms*3)    UP
Te0/0/0/0.506       2001:DB8::6:2
No                  n/a             0s(0s*0)          300ms(100ms*3)    UP
```

```
Te0/0/0/0.507       2001:DB8::7:2
No                  n/a             0s(0s*0)          300ms(100ms*3)    UP
Te0/0/0/0.508       2001:DB8::8:2
No                  n/a             0s(0s*0)          300ms(100ms*3)    UP
Te0/0/0/0.509       2001:DB8::9:2
No                  n/a             0s(0s*0)          300ms(100ms*3)    UP
```

In this sample output, **No** indicates that CPU offload is enabled and hardware offload is disabled.

BFD Object Tracking

Object Tracking is enhanced to support BFD to track the reachability of remote IP addresses.This will enable complete detection and HSRP switch over to happen within a time of less than one second as BFD can perform the detection in the order of few milliseconds

Configuring BFD Object Tracking:

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**track** *track-name* **Example:** `RP/0/RP0/CPU0:router(config)# track track1`	Enters track configuration mode. • *track-name*—Specifies a name for the object to be tracked.
Step 3	**type bfdrtr rate** *tx-rate* **Example:** `RP/0/RP0/CPU0:router(config-track)# type bfdrtr rate 4`	tx_rate - time in msec at which the BFD should probe the remote entity
Step 4	**debounce***debounce* **Example:** `RP/0/RP0/CPU0:router(config-if)# debounce 10`	debounce - count of consecutive BFD probes whose status should match before BFD notifies OT
Step 5	**interface** *if-name* **Example:**	if_name - interface name on the source to be used by BFD to check the remote BFD status.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-track-line-prot)# interface GigabitEthernet0/0/0/4`	
Step 6	**destaddress** *dest_addr* **Example:** `RP/0/RP0/CPU0:router(config-if)#destaddress 1.2.3.4`	dest_addr - IPV4 address of the remote BFD entity being tracked.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

BFD Dampening

Table 30: Feature History Table

Feature Name	Release	Description
IPv6 BFD dampening on Cisco 5700 routers	Release 25.1.1	Introduced in this release on: NCS 5700 fixed port routers; NCS 5700 line cards [Mode: Compatibility; Native] You can now prevent excessive network instability and connectivity fluctuations by enabling BFD dampening on IPv6 sessions. With extending support for IPv6 in addition to IPv4, notifications on unnecessary routing updates and traffic disruptions during link flapping are minimized. This mechanism helps in overall network stability and improves router performance.

Bidirectional Forwarding Detection (BFD) is a mechanism used by routing protocols to quickly realize and communicate the reachability failures to their neighbors. When BFD detects a reachability status change of a client, its neighbors are notified immediately. Sometimes it might be critical to minimize changes in routing tables so as not to impact convergence, in case of a micro failure. An unstable link that flaps excessively can cause other devices in the network to consume substantial processing resources, and that can cause routing protocols to lose synchronization with the state of the flapping link.

The BFD Dampening feature introduces a configurable exponential delay mechanism. This mechanism is designed to suppress the excessive effect of remote node reachability events flapping with BFD. The BFD Dampening feature allows the network operator to automatically dampen a given BFD session to prevent excessive notification to BFD clients, thus preventing unnecessary instability in the network. Dampening the notification to a BFD client suppresses BFD notification until the time the session under monitoring stops flapping and becomes stable.

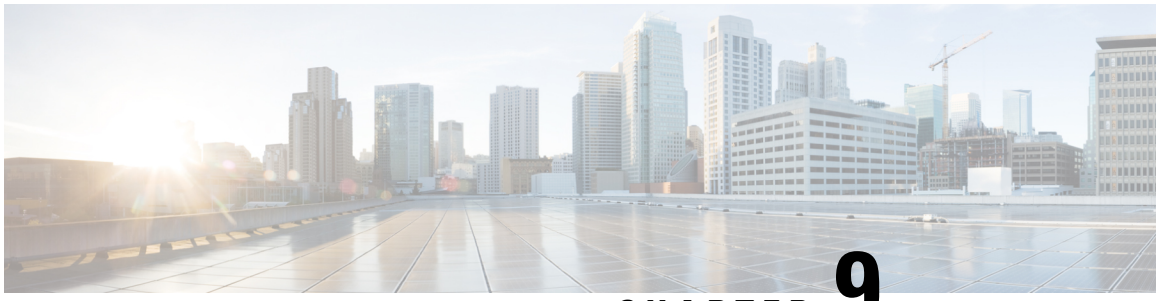
Configuring the BFD Dampening feature, especially on a high-speed interface with routing clients, improves convergence time and stability throughout the network. BFD dampening can be applied to all types of BFD sessions, including IPv4/single-hop, Multiprotocol Label Switching-Transport Profile (MPLS-TP), and Pseudo Wire (PW) Virtual Circuit Connection Verification (VCCV).

BFD Session Dampening

You can configure the BFD Dampening feature at the BFD template level (single-hop template). Dampening is applied to all the sessions that use the BFD template. If you choose not to have a session to be dampened, you should use a new BFD template without dampening for a new session.

Configuration Verification

```
/* Verify if the BFD session is up, and the timers are configured. */
RP/0//CPU0:router# show bfd session
Thu Jan 4 03:07:15.529 UTC
Interface Dest Addr Local det time(int*mult) State Echo Async H/W NPU
-------------- ---------- ------------------------ -------- ---- ----- ---
----
Te0/0/0/16.1 172.16.0.1 0s(0s*0) 200ms(100ms*2) UP Yes
```



CHAPTER 9

ECMP vs. UCMP Load Balancing

Load balancing is a forwarding mechanism that distributes traffic over multiple links based on certain parameters. Equal Cost Multi Path (ECMP) is a forwarding mechanism for routing packets along multiple paths of equal cost with the goal to achieve almost equally distributed link load sharing. This significantly impacts a router's next-hop (path) decision.

In ECMP, it is assumed that all links available are of similar speed which inherently means that the hash values that are computed are equally shared over the multiple paths available.

For instance, if we have two paths available, the buckets (which in the end identify the links to be chosen) will be assigned in a 50% / 50% loadsharing. This can be problematic when one path is say a 10G link and the other link is a 1G link. In this case, you probably want to assign a (near) 90/10 type deviation, but considering that BGP is not bandwidth aware, the 10G path is still chosen 50% of the time as much as the 1G link. In this scenario, not all paths are of equal cost path.

What UCMP does in this case is apply a *weight* to a path which means that we are giving more hash buckets to one path that has a higher weight. The weight applied is *static* in the sense that it is derived by the DMZ bandwidth extended community either assigned to a peer or as configured via the Route Policy Language (RPL) route manipulation functionality.

In general, a routing protocol decides a best path to a destination based on a metric. This metric is generally driven by the bandwidth of the circuit. When we have 3 paths available, say 1G/10G/100G, routing protocols generally discard the 1G/10G paths available. In defined cases, one may want to spread the load over the circuits based on the load they can carry. In this example, one may want to distribute traffic in a 1%/10%/89% fashion over the 1G/10G/100G paths available.

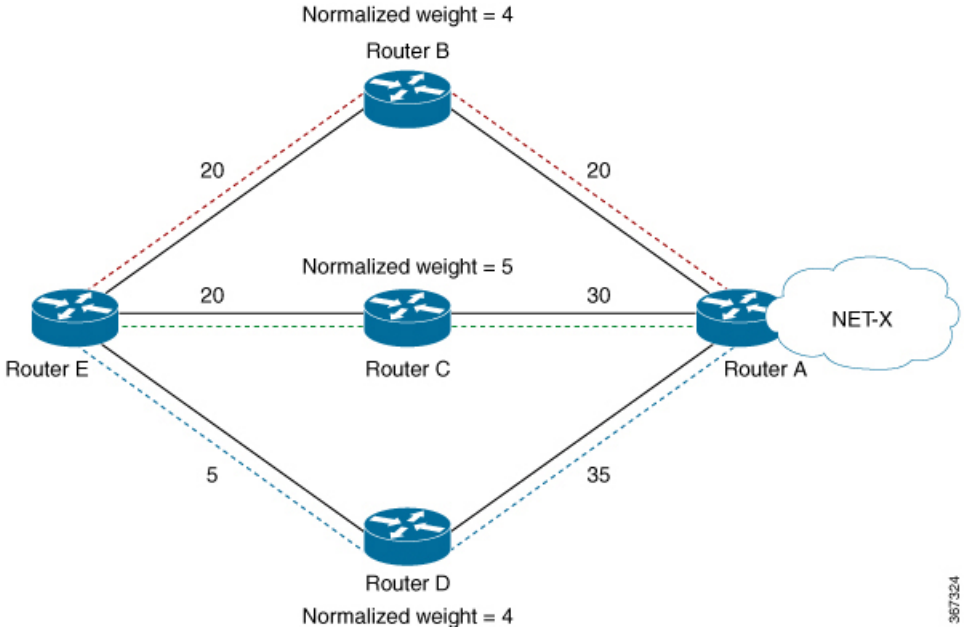
- UCMP Minimum Integer Ratio, on page 309
- Configuring IS-IS With Weight, on page 310
- Configuring IS-IS With Metric, on page 311
- Configuring BGP With Weights, on page 312
- Configuring TE Tunnel With Weights, on page 313
- Policy-Based Tunnel Selection, on page 314
- Interior Gateway Protocol (IGP) Destination-based Load Balancing (DLB), on page 326

UCMP Minimum Integer Ratio

The UCMP Minimum Integer Ratio feature saves hardware resources when programming UCMP, by using optimized number of buckets.

To calculate the UCMP minimum integer ratio, find the greatest common divisor (GCD) and divide all the calculated normalized weights.

In the following Figure, we have three configured weights 40, 50, and 40, with GCD as 10. To calculate the normalized weight, divide the configured weight by GCD. In this example, we need to divide 40 by 10, 50 by 10, and 40 by 10, which is 4, 5, and 4 respectively. Therefore 4, 5, and 4 are the new normalized weights.



New normalized weights are: 40/10 = 4, 50/10 = 5, and 40/10 = 4

If GCD is 1, then Normalized Weight = (Path weight/Total weight) * Maximum bucket size

Configuring IS-IS With Weight

The following example shows the IS-IS weight configuration with IPv4. The same can be done for IPv6, with or without SR.

```
CPU0:router(config)# router isis 1
RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet0/3/0/8
RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-isis-if-af)# weight 200
RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet0/3/0/9
RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-isis-if-af)# weight 300
```

Verification

The following example verifies CEF entry. Then, for two paths with weights of 200 and 300 respectively, and GCD of 100; the expected normalized weights are 2 and 3.

```
Router# show cef ipv4 97.0.0.0 detail

97.0.0.0/24, version 537, internal 0x1000001 0x0 (ptr 0x71bcaee0) [1], 0x0 (0x71b98870),
0x0 (0x0)
 Updated Oct 16 06:34:46.197
```

```
remote adjacency to GigabitEthernet0/3/0/8
Prefix Len 24, traffic index 0, precedence n/a, priority 2
 gateway array (0x71a6de10) reference count 13, flags 0x0, source rib (7), 0 backups
               [14 type 3 flags 0x8401 (0x71b02d90) ext 0x0 (0x0)]
 LW-LDI[type=3, refc=1, ptr=0x71b98870, sh-ldi=0x71b02d90]
 gateway array update type-time 1 Oct 16 06:34:46.196
LDI Update time Oct 16 06:34:46.197
LW-LDI-TS Oct 16 06:34:46.197
  via 1.0.0.2/32, GigabitEthernet0/3/0/8, 4 dependencies, weight 200, class 0 [flags 0x0]
   path-idx 0 NHID 0x0 [0x7244d2a4 0x0]
   next hop 1.0.0.2/32
   remote adjacency
  via 2.0.0.2/32, GigabitEthernet0/3/0/9, 4 dependencies, weight 300, class 0 [flags 0x0]
   path-idx 1 NHID 0x0 [0x7244d2f8 0x0]
   next hop 2.0.0.2/32
   remote adjacency

   Weight distribution:
   slot 0, weight 200, normalized_weight 2, class 0
   slot 1, weight 300, normalized_weight 3, class 0

   Load distribution: 0 1 0 1 1 (refcount 14)

   Hash  OK  Interface                 Address
   0     Y   GigabitEthernet0/3/0/8    remote
   1     Y   GigabitEthernet0/3/0/9    remote
   2     Y   GigabitEthernet0/3/0/8    remote
   3     Y   GigabitEthernet0/3/0/9    remote
   4     Y   GigabitEthernet0/3/0/9    remote
```

Configuring IS-IS With Metric

The following example shows IS-IS metric configuration with IPv4. The same can be done with IPv6.

```
Router# enable
RP/0/RSP0/CPU0:router(config)# router isis 1
RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet0/3/0/8
RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-isis-if-af)# metric 1
RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet0/3/0/9
RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-isis-if-af)# metric 100
```

Verification

The following example verifies CEF entry, and checks for the two paths with metric values of 1 and 100, respectively. In this example, the best path route metric is 21 and the UCMP path route metric is 120. Therefore, the calculation is as follows:

The best path route metric, 21 = (1 configured + 20 added by IS-IS), weight 0xFFFFFFFF (4294967295)

The UCMP path route metric, 120 = (100 + 20), weight = (21/120) * 4294967295 = 751619276

GCD is one. So Normalized Weight is:

(4294967295 * 64) / (4294967295 + 751619276) = 54

(751619276 * 64) / (4294967295 + 751619276) = 9

```
Router# show cef ipv4 97.0.0.0 detail
```

```
97.0.0.0/24, version 773, internal 0x1000001 0x0 (ptr 0x71bcaee0) [1], 0x0 (0x71b98870),
0x0 (0x0)
 Updated Oct 16 06:36:08.632
 remote adjacency to GigabitEthernet0/3/0/8
 Prefix Len 24, traffic index 0, precedence n/a, priority 2
  gateway array (0x71a6d9d0) reference count 2, flags 0x0, source rib (7), 0 backups
                [3 type 3 flags 0x8401 (0x71b02b90) ext 0x0 (0x0)]
  LW-LDI [type=3, refc=1, ptr=0x71b98870, sh-ldi=0x71b02b90]
  gateway array update type-time 1 Oct 16 06:36:08.632
 LDI Update time Oct 16 06:36:08.632
 LW-LDI-TS Oct 16 06:36:08.632
   via 1.0.0.2/32, GigabitEthernet0/3/0/8, 14 dependencies, weight 4294967295, class 0
[flags 0x0]
    path-idx 0 NHID 0x0 [0x7244d2a4 0x0]
    next hop 1.0.0.2/32
    remote adjacency
  via 2.0.0.2/32, GigabitEthernet0/3/0/9, 14 dependencies, weight 751619276, class 0 [flags
 0x0]
    path-idx 1 NHID 0x0 [0x7244d2f8 0x0]
    next hop 2.0.0.2/32
    remote adjacency


    Weight distribution:
    slot 0, weight 4294967295, normalized_weight 54, class 0
    slot 1, weight 751619276, normalized_weight 9, class 0
```

Configuring BGP With Weights

The following example shows BGP configuration with weights.

```
RP/0/RSP0/CPU0:router(config)# route-policy BW1
RP/0/RSP0/CPU0:router(config-rpl)# set extcommunity bandwidth (2906:45750000)
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
RP/0/RSP0/CPU0:router(config)# !
RP/0/RSP0/CPU0:router(config)# route-policy BW2
RP/0/RSP0/CPU0:router(config-rpl)# set extcommunity bandwidth (2906:47250000)
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
RP/0/RSP0/CPU0:router(config)# !
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)# !
RP/0/RSP0/CPU0:router(config)# router bgp 1
RP/0/RSP0/CPU0:router(config-bgp)# bgp bestpath as-path multipath-relax
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)# maximum-paths eibgp 64
RP/0/RSP0/CPU0:router(config-bgp-af)# !
RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 1.0.0.2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# ebgp-multihop 255
RP/0/RSP0/CPU0:router(config-bgp-nbr)# dmz-link-bandwidth
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# multipath
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy BW1 in
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# !
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# neighbor 2.0.0.2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# ebgp-multihop 255
RP/0/RSP0/CPU0:router(config-bgp-nbr)# dmz-link-bandwidth
```

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# multipath
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy BW2 in
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
```

Verification

Step 1: Verify CEF entry:

Via 1.0.0.2: set extcommunity bandwidth (2906:45750000) – Weight = 45750000/125=366000 (125 ratio because baud)

Via 2.0.0.2: set extcommunity bandwidth (2906:47250000) – Weight = 47250000/125=378000

GCD is 6, so norm_weight = 61 and 63. Though 61 + 63 > 64.

Step 2: GCD of weights 61 and 63 is 1. Therefore, Normalised Weight = (Path weight/Total weight) * Maximum bucket size. The maximum bucket size value is 64. Total weight = 61+63 = 124.

norm_weight1 = (61/124) * 64 = 31, norm_weight2 = (63/124) * 64 = 32

You can verify the weight distribution in BGP, using the following command:

```
Router # show cef vrf default ipv4 97.0.0.0 detail


97.0.0.0/24, version 1965, internal 0x5000001 0x0 (ptr 0x71bcb620) [1], 0x0 (0x0), 0x0 (0x0)
 Updated Oct 16 08:15:02.958
 Prefix Len 24, traffic index 0, precedence n/a, priority 4
  gateway array (0x72a5e2f8) reference count 10, flags 0x2010, source rib (7), 0 backups
                [1 type 3 flags 0x48441 (0x71b02cd0) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Oct 16 08:15:02.958
 LDI Update time Oct 16 08:15:02.959


    Weight distribution:
    slot 0, weight 366000, normalized_weight 31
    slot 1, weight 378000, normalized_weight 32


  Level 1 - Load distribution: 0 1 0 1 0 1 0

1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 1 0 1 0 1 0 1 0 1 1
  [0] via 1.0.0.2/32, recursive
  [1] via 2.0.0.2/32, recursive
```

Configuring TE Tunnel With Weights

Verification

In the following example, the weight is distributed among the five TE tunnels.

```
Router# show cef ipv4 97.0.0.0 detail
(…)
   via 200.0.0.1/32, tunnel-te1, 3 dependencies, weight 8, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x7244d2f8 0x0]
    next hop 200.0.0.1/32
    local adjacency
   via 200.0.0.1/32, tunnel-te2, 3 dependencies, weight 4, class 0 [flags 0x0]
```

```
   path-idx 1 NHID 0x0 [0x7244e948 0x0]
   next hop 200.0.0.1/32
   local adjacency
  via 200.0.0.1/32, tunnel-te3, 3 dependencies, weight 1, class 0 [flags 0x0]
   path-idx 2 NHID 0x0 [0x7244d544 0x0]
   next hop 200.0.0.1/32
   local adjacency
  via 200.0.0.1/32, tunnel-te4, 3 dependencies, weight 1, class 0 [flags 0x0]
   path-idx 3 NHID 0x0 [0x7244d694 0x0]
   next hop 200.0.0.1/32
   local adjacency
  via 200.0.0.1/32, tunnel-te5, 3 dependencies, weight 1, class 0 [flags 0x0]
   path-idx 4 NHID 0x0 [0x7244d7e4 0x0]
   next hop 200.0.0.1/32
   local adjacency

   Weight distribution:
   slot 0, weight 8, normalized_weight 8, class 0
   slot 1, weight 4, normalized_weight 4, class 0
   slot 2, weight 1, normalized_weight 1, class 0
   slot 3, weight 1, normalized_weight 1, class 0
   slot 4, weight 1, normalized_weight 1, class 0
```

Policy-Based Tunnel Selection

Policy-Based Tunnel Selection (PBTS) provides a mechanism that lets you direct traffic into specific TE tunnels based on different criteria.

PBTS is a special case in UCMP calculation. It uses load share command to configure weight. The UCMP algorithm normalizes each class independently and it uses max_path from PD specific max_tunnels_per_class. UCMP with PBTS can have more total_paths (buckets) than the supported number of paths (buckets) for all Forwarding Classes (FCs).

All other XR platform sets 8 buckets per FC and 64 buckets for all 8 (0-7) FCs. After normalization, the total number buckets do not exceed platform limit.

Example

The **show cef ipv6** command displays the PBTS class information in the following output.

```
Router# show cef ipv6 97:: detail

97::/64, version 88177, internal 0x1000001 0x0 (ptr 0x980eef7c) [1], 0x0 (0x974366b8), 0xa28
 (0x988842c0)
 Updated Mar  7 05:44:46.875

 Prefix Len 64, traffic index 0, precedence n/a, priority 2
  gateway array (0x97e54770) reference count 11, flags 0x28, source rib (7), 0 backups
                [12 type 1 flags 0x200401 (0x9799a3f8) ext 0x0 (0x0)]
  LW-LDI[type=1, refc=1, ptr=0x974366b8, sh-ldi=0x9799a3f8]
  gateway array update type-time 4 Mar  7 05:46:11.118
 LDI Update time Mar  7 05:46:11.118
 LW-LDI-TS Mar  7 05:46:11.118


   via ::ffff:200.0.0.1/128, tunnel-te45, 3 dependencies, weight 1, forward class 6 [flags
0x0]

    path-idx 0 NHID 0x0 [0x97b51978 0x0]
```

```
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te46, 3 dependencies, weight 1, forward class 6 [flags
0x0]

    path-idx 1 NHID 0x0 [0x97b51648 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te47, 3 dependencies, weight 1, forward class 6 [flags
0x0]

    path-idx 2 NHID 0x0 [0x97b51c20 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te48, 3 dependencies, weight 1, forward class 6 [flags
0x0]

    path-idx 3 NHID 0x0 [0x97b52308 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te49, 3 dependencies, weight 1, forward class 7 [flags
0x0]

    path-idx 4 NHID 0x0 [0x97b518f0 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te1, 3 dependencies, weight 3, forward class 1 [flags
0x0]

    path-idx 5 NHID 0x0 [0x97b4f338 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te2, 3 dependencies, weight 500, forward class 1 [flags
0x0]

    path-idx 6 NHID 0x0 [0x97b50328 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te3, 3 dependencies, weight 1, forward class 1 [flags
0x0]

    path-idx 7 NHID 0x0 [0x97b4ede8 0x0]
    next hop VRF - 'default', table - 0xe0000000
```

```
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

   via ::ffff:200.0.0.1/128, tunnel-te4, 3 dependencies, weight 1, forward class 1 [flags
0x0]

    path-idx 8 NHID 0x0 [0x97b4eb40 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

   via ::ffff:200.0.0.1/128, tunnel-te5, 3 dependencies, weight 1, forward class 1 [flags
0x0]

    path-idx 9 NHID 0x0 [0x97b4fff8 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

   via ::ffff:200.0.0.1/128, tunnel-te6, 3 dependencies, weight 1, forward class 1 [flags
0x0]

    path-idx 10 NHID 0x0 [0x97b4f778 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

   via ::ffff:200.0.0.1/128, tunnel-te7, 3 dependencies, weight 1, forward class 1 [flags
0x0]

    path-idx 11 NHID 0x0 [0x97b4f118 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

   via ::ffff:200.0.0.1/128, tunnel-te8, 3 dependencies, weight 1, forward class 1 [flags
0x0]

    path-idx 12 NHID 0x0 [0x97b4ee70 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

   via ::ffff:200.0.0.1/128, tunnel-te9, 3 dependencies, weight 1, forward class 2 [flags
0x0]

    path-idx 13 NHID 0x0 [0x97b4f090 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
    local adjacency
     labels imposed {ExpNullv6}

   via ::ffff:200.0.0.1/128, tunnel-te10, 3 dependencies, weight 1, forward class 2 [flags
 0x0]

    path-idx 14 NHID 0x0 [0x97b4f448 0x0]
    next hop VRF - 'default', table - 0xe0000000
    next hop ::ffff:200.0.0.1/128
```

```
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te11, 3 dependencies, weight 1, forward class 2 [flags
0x0]

   path-idx 15 NHID 0x0 [0x97b4faa8 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te12, 3 dependencies, weight 1, forward class 2 [flags
0x0]

   path-idx 16 NHID 0x0 [0x97b4f008 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te13, 3 dependencies, weight 1, forward class 2 [flags
0x0]

   path-idx 17 NHID 0x0 [0x97b50218 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te14, 3 dependencies, weight 1, forward class 2 [flags
0x0]

   path-idx 18 NHID 0x0 [0x97b4fbb8 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te15, 3 dependencies, weight 1, forward class 2 [flags
0x0]

   path-idx 19 NHID 0x0 [0x97b4ed60 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te16, 3 dependencies, weight 1, forward class 2 [flags
0x0]

   path-idx 20 NHID 0x0 [0x97b4fcc8 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te17, 3 dependencies, weight 1, forward class 3 [flags
0x0]


   path-idx 21 NHID 0x0 [0x97b50190 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
```

```
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te18, 3 dependencies, weight 1, forward class 3 [flags
0x0]


   path-idx 22 NHID 0x0 [0x97b4f998 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te19, 3 dependencies, weight 1, forward class 3 [flags
0x0]


   path-idx 23 NHID 0x0 [0x97b4fee8 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te20, 3 dependencies, weight 1, forward class 3 [flags
0x0]

   path-idx 24 NHID 0x0 [0x97b505d0 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te21, 3 dependencies, weight 1, forward class 3 [flags
0x0]

   path-idx 25 NHID 0x0 [0x97b4fc40 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te22, 3 dependencies, weight 1, forward class 3 [flags
0x0]

   path-idx 26 NHID 0x0 [0x97b50988 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te23, 3 dependencies, weight 1, forward class 3 [flags
0x0]

   path-idx 27 NHID 0x0 [0x97b50080 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}
```

```
  via ::ffff:200.0.0.1/128, tunnel-te24, 3 dependencies, weight 1, forward class 3 [flags
0x0]

   path-idx 28 NHID 0x0 [0x97b4fd50 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te25, 3 dependencies, weight 1, forward class 4 [flags
0x0]


   path-idx 29 NHID 0x0 [0x97b503b0 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te26, 3 dependencies, weight 1, forward class 4 [flags
0x0]


   path-idx 30 NHID 0x0 [0x97b507f0 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te27, 3 dependencies, weight 1, forward class 4 [flags
0x0]

   path-idx 31 NHID 0x0 [0x97b4ff70 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te28, 3 dependencies, weight 1, forward class 4 [flags
0x0]


   path-idx 32 NHID 0x0 [0x97b50548 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te29, 3 dependencies, weight 1, forward class 4 [flags
0x0]

   path-idx 33 NHID 0x0 [0x97b4fb30 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}
```

```
  via ::ffff:200.0.0.1/128, tunnel-te30, 3 dependencies, weight 1, forward class 4 [flags
0x0]

   path-idx 34 NHID 0x0 [0x97b506e0 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te31, 3 dependencies, weight 1, forward class 4 [flags
0x0]

   path-idx 35 NHID 0x0 [0x97b51208 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te32, 3 dependencies, weight 1, forward class 4 [flags
0x0]

   path-idx 36 NHID 0x0 [0x97b502a0 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te33, 3 dependencies, weight 1, forward class 5 [flags
0x0]

   path-idx 37 NHID 0x0 [0x97b514b0 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te34, 3 dependencies, weight 1, forward class 5 [flags
0x0]

   path-idx 38 NHID 0x0 [0x97b50c30 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te35, 3 dependencies, weight 1, forward class 5 [flags
0x0]

   path-idx 39 NHID 0x0 [0x97b50b20 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te36, 3 dependencies, weight 1, forward class 5 [flags
0x0]

   path-idx 40 NHID 0x0 [0x97b50cb8 0x0]
```

```
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te37, 3 dependencies, weight 1, forward class 5 [flags
0x0]


   path-idx 41 NHID 0x0 [0x97b51180 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te38, 3 dependencies, weight 1, forward class 5 [flags
0x0]

   path-idx 42 NHID 0x0 [0x97b51428 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te39, 3 dependencies, weight 1, forward class 5 [flags
0x0]


   path-idx 43 NHID 0x0 [0x97b51758 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te40, 3 dependencies, weight 1, forward class 5 [flags
0x0]

   path-idx 44 NHID 0x0 [0x97b520e8 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te41, 3 dependencies, weight 1, forward class 6 [flags
0x0]

   path-idx 45 NHID 0x0 [0x97b51538 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te42, 3 dependencies, weight 1, forward class 6 [flags
0x0]

   path-idx 46 NHID 0x0 [0x97b50dc8 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
```

```
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te43, 3 dependencies, weight 1, forward class 6 [flags
0x0]

   path-idx 47 NHID 0x0 [0x97b51b10 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te44, 3 dependencies, weight 1, forward class 6 [flags
0x0]

   path-idx 48 NHID 0x0 [0x97b516d0 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te50, 3 dependencies, weight 1, forward class 7 [flags
0x0]

   path-idx 49 NHID 0x0 [0x97b525b0 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te51, 3 dependencies, weight 1, forward class 7 [flags
0x0]

   path-idx 50 NHID 0x0 [0x97b52638 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te52, 3 dependencies, weight 1, forward class 7 [flags
0x0]

   path-idx 51 NHID 0x0 [0x97b51f50 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te53, 3 dependencies, weight 1, forward class 7 [flags
0x0]

   path-idx 52 NHID 0x0 [0x97b52060 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te54, 3 dependencies, weight 1, forward class 7 [flags
0x0]
```

```
   path-idx 53 NHID 0x0 [0x97b527d0 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te55, 3 dependencies, weight 1, forward class 7 [flags
0x0]

   path-idx 54 NHID 0x0 [0x97b52280 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}


  via ::ffff:200.0.0.1/128, tunnel-te56, 3 dependencies, weight 1, forward class 7 [flags
0x0]

   path-idx 55 NHID 0x0 [0x97b52d20 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te57, 3 dependencies, weight 1, class 0 [flags 0x0]

   path-idx 56 NHID 0x0 [0x97b51ca8 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te58, 3 dependencies, weight 1, class 0 [flags 0x0]

   path-idx 57 NHID 0x0 [0x97b52858 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te59, 3 dependencies, weight 1, class 0 [flags 0x0]

   path-idx 58 NHID 0x0 [0x97b52390 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te60, 3 dependencies, weight 1, class 0 [flags 0x0]

   path-idx 59 NHID 0x0 [0x97b52a78 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}

  via ::ffff:200.0.0.1/128, tunnel-te61, 3 dependencies, weight 1, class 0 [flags 0x0]

   path-idx 60 NHID 0x0 [0x97b52c10 0x0]
   next hop VRF - 'default', table - 0xe0000000
   next hop ::ffff:200.0.0.1/128
   local adjacency
    labels imposed {ExpNullv6}
```

```
via ::ffff:200.0.0.1/128, tunnel-te62, 3 dependencies, weight 1, class 0 [flags 0x0]

 path-idx 61 NHID 0x0 [0x97b52da8 0x0]
 next hop VRF - 'default', table - 0xe0000000
 next hop ::ffff:200.0.0.1/128
 local adjacency
  labels imposed {ExpNullv6}


via ::ffff:200.0.0.1/128, tunnel-te63, 3 dependencies, weight 1, class 0 [flags 0x0]

 path-idx 62 NHID 0x0 [0x97b52c98 0x0]
 next hop VRF - 'default', table - 0xe0000000
 next hop ::ffff:200.0.0.1/128
 local adjacency
  labels imposed {ExpNullv6}


 Weight distribution:

 slot 0, weight 1, normalized_weight 1, class 0
 slot 1, weight 1, normalized_weight 1, class 0
 slot 2, weight 1, normalized_weight 1, class 0
 slot 3, weight 1, normalized_weight 1, class 0
 slot 4, weight 1, normalized_weight 1, class 0
 slot 5, weight 1, normalized_weight 1, class 0
 slot 6, weight 1, normalized_weight 1, class 0
 slot 7, weight 1, normalized_weight 1, forward class 1
 slot 8, weight 3, normalized_weight 1, forward class 1
 slot 9, weight 500, normalized_weight 1, forward class 1
 slot 10, weight 1, normalized_weight 1, forward class 1
 slot 11, weight 1, normalized_weight 1, forward class 1
 slot 12, weight 1, normalized_weight 1, forward class 1
 slot 13, weight 1, normalized_weight 1, forward class 1
 slot 14, weight 1, normalized_weight 1, forward class 1
 slot 15, weight 1, normalized_weight 1, forward class 2
 slot 16, weight 1, normalized_weight 1, forward class 2
 slot 17, weight 1, normalized_weight 1, forward class 2
 slot 18, weight 1, normalized_weight 1, forward class 2
 slot 19, weight 1, normalized_weight 1, forward class 2
 slot 20, weight 1, normalized_weight 1, forward class 2
 slot 21, weight 1, normalized_weight 1, forward class 2
 slot 22, weight 1, normalized_weight 1, forward class 2
 slot 23, weight 1, normalized_weight 1, forward class 3
 slot 24, weight 1, normalized_weight 1, forward class 3
 slot 25, weight 1, normalized_weight 1, forward class 3
 slot 26, weight 1, normalized_weight 1, forward class 3
 slot 27, weight 1, normalized_weight 1, forward class 3
 slot 28, weight 1, normalized_weight 1, forward class 3
 slot 29, weight 1, normalized_weight 1, forward class 3
 slot 30, weight 1, normalized_weight 1, forward class 3
 slot 31, weight 1, normalized_weight 1, forward class 4
 slot 32, weight 1, normalized_weight 1, forward class 4
 slot 33, weight 1, normalized_weight 1, forward class 4
 slot 34, weight 1, normalized_weight 1, forward class 4
 slot 35, weight 1, normalized_weight 1, forward class 4
 slot 36, weight 1, normalized_weight 1, forward class 4
 slot 37, weight 1, normalized_weight 1, forward class 4
 slot 38, weight 1, normalized_weight 1, forward class 4
 slot 39, weight 1, normalized_weight 1, forward class 5
 slot 40, weight 1, normalized_weight 1, forward class 5
 slot 41, weight 1, normalized_weight 1, forward class 5
 slot 42, weight 1, normalized_weight 1, forward class 5
```

```
    slot 43, weight 1, normalized_weight 1, forward class 5
    slot 44, weight 1, normalized_weight 1, forward class 5
    slot 45, weight 1, normalized_weight 1, forward class 5
    slot 46, weight 1, normalized_weight 1, forward class 5
    slot 47, weight 1, normalized_weight 1, forward class 6
    slot 48, weight 1, normalized_weight 1, forward class 6
    slot 49, weight 1, normalized_weight 1, forward class 6
    slot 50, weight 1, normalized_weight 1, forward class 6
    slot 51, weight 1, normalized_weight 1, forward class 6
    slot 52, weight 1, normalized_weight 1, forward class 6
    slot 53, weight 1, normalized_weight 1, forward class 6
    slot 54, weight 1, normalized_weight 1, forward class 6
    slot 55, weight 1, normalized_weight 1, forward class 7
    slot 56, weight 1, normalized_weight 1, forward class 7
    slot 57, weight 1, normalized_weight 1, forward class 7
    slot 58, weight 1, normalized_weight 1, forward class 7
    slot 59, weight 1, normalized_weight 1, forward class 7
    slot 60, weight 1, normalized_weight 1, forward class 7
    slot 61, weight 1, normalized_weight 1, forward class 7
    slot 62, weight 1, normalized_weight 1, forward class 7

    PBTS class information:

      class 0: 7 paths, offset 0

      forward class 1: 8 paths, offset 7
      forward class 2: 8 paths, offset 15
      forward class 3: 8 paths, offset 23
      forward class 4: 8 paths, offset 31
      forward class 5: 8 paths, offset 39
      forward class 6: 8 paths, offset 47
      forward class 7: 8 paths, offset 55

    Load distribution: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 (refcount 12)


    Hash  OK  Interface                Address

    0     Y   tunnel-te57              point2point
    1     Y   tunnel-te58              point2point
    2     Y   tunnel-te59              point2point
    3     Y   tunnel-te60              point2point
    4     Y   tunnel-te61              point2point
    5     Y   tunnel-te62              point2point
    6     Y   tunnel-te63              point2point
    7     Y   tunnel-te8               point2point
    8     Y   tunnel-te1               point2point
    9     Y   tunnel-te2               point2point
    10    Y   tunnel-te3               point2point


    11    Y   tunnel-te4               point2point
    12    Y   tunnel-te5               point2point
    13    Y   tunnel-te6               point2point
    14    Y   tunnel-te7               point2point
    15    Y   tunnel-te16              point2point
    16    Y   tunnel-te9               point2point
    17    Y   tunnel-te10              point2point
    18    Y   tunnel-te11              point2point
    19    Y   tunnel-te12              point2point
    20    Y   tunnel-te13              point2point
    21    Y   tunnel-te14              point2point
```

```
22    Y   tunnel-te15              point2point
23    Y   tunnel-te24              point2point
24    Y   tunnel-te17              point2point
25    Y   tunnel-te18              point2point
26    Y   tunnel-te19              point2point
27    Y   tunnel-te20              point2point
28    Y   tunnel-te21              point2point
29    Y   tunnel-te22              point2point
30    Y   tunnel-te23              point2point
31    Y   tunnel-te32              point2point
32    Y   tunnel-te25              point2point
33    Y   tunnel-te26              point2point
34    Y   tunnel-te27              point2point
35    Y   tunnel-te28              point2point
36    Y   tunnel-te29              point2point
37    Y   tunnel-te30              point2point
38    Y   tunnel-te31              point2point
39    Y   tunnel-te40              point2point
40    Y   tunnel-te33              point2point
41    Y   tunnel-te34              point2point
42    Y   tunnel-te35              point2point
43    Y   tunnel-te36              point2point
44    Y   tunnel-te37              point2point
45    Y   tunnel-te38              point2point
46    Y   tunnel-te39              point2point
47    Y   tunnel-te44              point2point
48    Y   tunnel-te45              point2point
49    Y   tunnel-te46              point2point
50    Y   tunnel-te47              point2point
51    Y   tunnel-te48              point2point
52    Y   tunnel-te41              point2point
53    Y   tunnel-te42              point2point
54    Y   tunnel-te43              point2point
55    Y   tunnel-te56              point2point
56    Y   tunnel-te49              point2point
57    Y   tunnel-te50              point2point
58    Y   tunnel-te51              point2point
59    Y   tunnel-te52              point2point
60    Y   tunnel-te53              point2point
61    Y   tunnel-te54              point2point
62    Y   tunnel-te55              point2point
```

Interior Gateway Protocol (IGP) Destination-based Load Balancing (DLB)

Currently, the router supports upto 2K labelled prefixes with Equal Cost Multi Path (ECMP). From Cisco IOS XR Software Release 7.1.1, with the introduction of the Interior Gateway Protocol (IGP) Destination-based Load Balancing (DLB) feature, the router can support higher scale of labelled prefixes.

If this feature is enabled, the software selects one path out of the available multipaths based on a software hash of the destination IP address and then programs the data plane with the selected single path. Thus, the traffic gets distributed on available paths by pre-selection of the path based on the per-prefix hash.

However, if Layer 2 Virtual Private Network (L2VPN) or Border Gateway Protocol (BGP) service uses a IGP DLB prefix for next-hop reachability, then the software removes DLB from that particular prefix and it behaves as if L2VPN or BGP runs on a non-DLB IGP.

Restrictions for IGP DLB

The following are the restrictions for IGP DLB:

- IGP DLB is not enabled in software, by default.
- ECMP resource utilization happens only on enabling BGP or L2 services for the DLB prefix. However disabling BGP or L2 services for the prefix does not release the ECMP resource immediately. ECMP resources get released only when the prefix is re-learned again or with a router reload.
- For the configuration command **hw-module fib dlb level-1 enable** to take effect, you should reload the LC.

Configuring IGP DLB

Without IGP DLB enabled, all prefixes consume ECMP hardware resources.

However, with IGP DLB enabled, DLB IP and labeled prefixes do not consume ECMP resources. Only BGP and L2VPN services over a DLB IGP prefix consume ECMP resources. This enables the router to handle higher prefix scale by the efficient use of the ECMP hardware resources.

The following example shows the ECMP software-based load-balancing configuration:

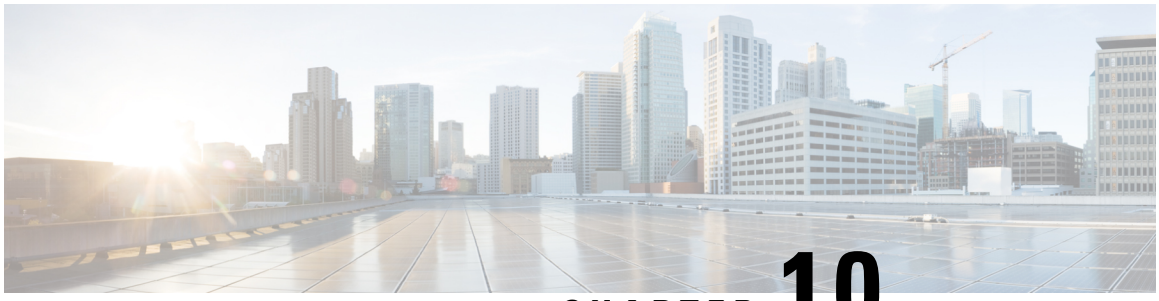
```
Router# config terminal
Router(config)# hw-module fib dlb level-1 enable
Router(config)# commit
Router(config)# end
Router# reload location all
Proceed with reload? [confirm]
```

Verification

In the below command, resource usage will be less when IGP DLB is enabled:

```
Router# show controllers npu resources ecmpfec location 0/3/CPU0
HW Resource Information
    Name                            : ecmp_fec
    Asic Type                       : Jericho Plus

NPU-0
OOR Summary
        Estimated Max Entries       : 4096
        Red Threshold               : 95
        Yellow Threshold            : 80
        OOR State                   : Green


Current Usage
        Total In-Use                : 3        (0 %)
        ipnhgroup                   : 3        (0 %)
        ip6nhgroup                  : 0        (0 %)


NPU-1
OOR Summary
        Estimated Max Entries       : 4096
        Red Threshold               : 95
        Yellow Threshold            : 80
```

```
        OOR State                   : Green


Current Usage
        Total In-Use                : 3        (0 %)
        ipnhgroup                   : 3        (0 %)
        ip6nhgroup                  : 0        (0 %)


NPU-2
OOR Summary
        Estimated Max Entries       : 4096
        Red Threshold               : 95
        Yellow Threshold            : 80
        OOR State                   : Green


Current Usage
        Total In-Use                : 3        (0 %)
        ipnhgroup                   : 3        (0 %)
        ip6nhgroup                  : 0        (0 %)


NPU-3
OOR Summary
        Estimated Max Entries       : 4096
        Red Threshold               : 95
        Yellow Threshold            : 80
        OOR State                   : Green


Current Usage
        Total In-Use                : 3        (0 %)
        ipnhgroup                   : 3        (0 %)
        ip6nhgroup                  : 0        (0 %)

Router# show cef 209.165.200.225 hardware egress detail location npu0
….
HW Walk:
LEAF:
    PI:0x308bec58b8 PD:0x308bec5958 rev:4910 type: IPV4 (0)
    LEAF location: LEM
    FEC key: 0x20e20000e0b

    LWLDI:
        EOS0/1 LDI: ECD_MARKED  SERVICE_MARKED  # shows ECMP resource usage with IGP DLB
        LSP pattern:3
```



CHAPTER 10

Implementing Fast Reroute Loop-Free Alternate

Fast Reroute Loop-Free Alternate feature enables you to tunnel a packet around a failed link to a remote loop-free alternate that is more than one hop away.

- Prerequisites for Fast Reroute with Loop-Free Alternate, on page 329
- Restrictions for Fast Reroute with Loop-Free Alternate, on page 329
- IS-IS and FRR, on page 330
- Repair Paths, on page 330
- LFA Overview, on page 330
- LFA Calculation, on page 331
- Interaction Between RIB and Routing Protocols, on page 331
- Fast Reroute with Remote Loop-Free Alternate, on page 332
- Configuration , on page 333

Prerequisites for Fast Reroute with Loop-Free Alternate

- Fast Reroute with Loop-Free Alternate functionality can protect paths that are reachable through an interface only if the interface is a point-to-point interface.
- When a LAN interface is physically connected to a single neighbor, you should configure the LAN interface as a point-to-point interface so that it can be protected through Loop-Free Alternate (LFA) FRR.
- For a proper deployment for Fast Reroute with Remote Loop-Free Alternate feature, the protected link should also be configured with BFD

Restrictions for Fast Reroute with Loop-Free Alternate

- Load balance support is available for FRR-protected prefixes, but the 50 ms cutover time is not guaranteed.
- A maximum of eight FRR-protected interfaces can simultaneously undergo a cutover.
- LFA calculations are restricted to interfaces or links belonging to the same level or area. Hence, excluding all neighbors on the same LAN when computing the backup LFA can result in repairs being unavailable in a subset of topologies.
- Only physical and physical port-channel interfaces and subinterfaces are protected. Tunnels and virtual interfaces are not protected.

- The remote LFA backup path for MPLS traffic can be setup only using LDP. Only per-prefix protection is supported.
- Border Gateway Protocol (BGP) Prefix-Independent Convergence (PIC) and FRR can be configured on the same interface as long as they are not used for the same prefix.

IS-IS and FRR

When a local link fails in a network, IS-IS recomputes new primary next-hop routes for all affected prefixes. These prefixes are updated in the RIB and the Forwarding Information Base (FIB). Until the primary prefixes are updated in the forwarding plane, traffic directed towards the affected prefixes are discarded. This process can take hundreds of milliseconds.

In FRR, IS-IS computes LFA next-hop routes for the forwarding plane to use in case of primary path failures. LFA is computed per prefix.

When there are multiple LFAs for a given primary path, IS-IS uses a tiebreaking rule to pick a single LFA for a primary path. In case of a primary path with multiple LFA paths, prefixes are distributed equally among LFA paths.

Repair Paths

Repair paths forward traffic during a routing transition. When a link or a router fails, due to the loss of a physical layer signal, initially, only the neighboring routers are aware of the failure. All other routers in the network are unaware of the nature and location of this failure until information about this failure is propagated through a routing protocol, which may take several hundred milliseconds. It is, therefore, necessary to arrange for packets affected by the network failure to be steered to their destinations.

A router adjacent to the failed link employs a set of repair paths for packets that would have used the failed link. These repair paths are used from the time the router detects the failure until the routing transition is complete. By the time the routing transition is complete, all routers in the network revise their forwarding data and the failed link is eliminated from the routing computation.

Repair paths are precomputed in anticipation of failures so that they can be activated the moment a failure is detected.

The LFA FRR feature uses the following repair paths:

- Equal Cost Multipath (ECMP) uses a link as a member of an equal cost path-split set for a destination. The other members of the set can provide an alternative path when the link fails.
- LFA is a next-hop route that delivers a packet to its destination without looping back. Downstream paths are a subset of LFAs.

LFA Overview

LFA is a node other than the primary neighbor. Traffic is redirected to an LFA after a network failure. An LFA makes the forwarding decision without any knowledge of the failure.

An LFA must neither use a failed element nor use a protecting node to forward traffic. An LFA must not cause loops. By default, LFA is enabled on all supported interfaces as long as the interface can be used as a primary path.

Advantages of using per-prefix LFAs are as follows:

- The repair path forwards traffic during transition when the primary path link is down.
- All destinations having a per-prefix LFA are protected. This leaves only a subset (a node at the far side of the failure) unprotected.

LFA Calculation

The general algorithms to compute per-prefix LFAs can be found in RFC 5286. IS-IS implements RFC 5286 with a small change to reduce memory usage. Instead of performing a Shortest Path First (SPF) calculation for all neighbors before examining prefixes for protection, IS-IS examines prefixes after SPF calculation is performed for each neighbor. Because IS-IS examines prefixes after SPF calculation is performed, IS-IS retains the best repair path after SPF calculation is performed for each neighbor. IS-IS does not have to save SPF results for all neighbors.

Interaction Between RIB and Routing Protocols

A routing protocol computes repair paths for prefixes by implementing tiebreaking algorithms. The end result of the computation is a set of prefixes with primary paths, where some primary paths are associated with repair paths.

A tiebreaking algorithm considers LFAs that satisfy certain conditions or have certain attributes. When there is more than one LFA, configure the **fast-reroute per-prefix** command with the **tie-break** keyword. If a rule eliminates all candidate LFAs, then the rule is skipped.

A primary path can have multiple LFAs. A routing protocol is required to implement default tiebreaking rules and to allow you to modify these rules. The objective of the tiebreaking algorithm is to eliminate multiple candidate LFAs, select one LFA per primary path per prefix, and distribute the traffic over multiple candidate LFAs when the primary path fails.

Tiebreaking rules cannot eliminate all candidates.

The following attributes are used for tiebreaking:

- Downstream—Eliminates candidates whose metric to the protected destination is lower than the metric of the protecting node to the destination.
- Linecard-disjoint—Eliminates candidates sharing the same linecard with the protected path.
- Shared Risk Link Group (SRLG)—Eliminates candidates that belong to one of the protected path SRLGs.
- Load-sharing—Distributes remaining candidates among prefixes sharing the protected path.
- Lowest-repair-path-metric—Eliminates candidates whose metric to the protected prefix is higher.
- Node protecting—Eliminates candidates that are not node protected.
- Primary-path—Eliminates candidates that are not ECMPs.

- Secondary-path—Eliminates candidates that are ECMPs.

Fast Reroute with Remote Loop-Free Alternate

Fast Reroute with Remote Loop-Free Alternate (FRR Remote LFA) feature enables you to tunnel a packet around a failed link to a remote loop-free alternate that is more than one hop away.

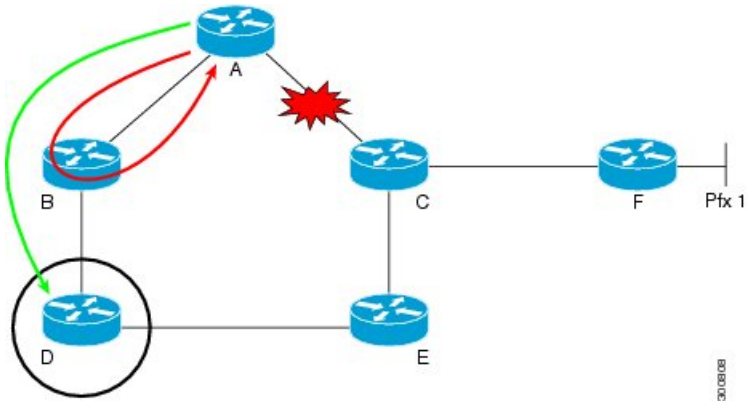
When a link or a router fails, distributed routing algorithms compute new routes that take into account the failure. The time taken for computation is called routing transition. Until the transition is complete and all routers are converged on a common view of the network, the connectivity between the source and destination pairs is interrupted. You can use the IP Loop-Free Alternate (LFA) Fast Reroute (FRR) to reduce the routing transition time to less than 50 milliseconds using a precomputed alternate next hop. When a router is notified of a link failure, the router immediately switches over to the repair path to reduce traffic loss. Note that the routing transition in IGP/BGP convergence can take up to several hundreds of milliseconds.

IP Loop-Free Alternate (LFA) Fast Reroute (FRR) supports the precomputation of repair paths. Intermediate System-to-Intermediate System (IS-IS) routing protocol enables the repair path computation. The resulting repair paths are sent to the Routing Information Base (RIB). Cisco Express Forwarding (formerly known as CEF) and Open Shortest Path First (OSPF) installs the repair path.

With IP local LFA FRR, IGPs only compute directly connected neighbor as an LFA backup path to protect the given prefix's primary path. Label Distribution Protocol (LDP) sets up labeled backup LSP with the next-hop for the protected prefix. Some topologies (for example the commonly used ring-based topology) require protection that is not afforded by LFA FRR. In such cases, use the LDP-based FRR Remote LFA feature where IGPs compute non-directly connected neighbor, which are more than one hop away, as LFA backup path to protect the given prefix's primary path. The LDP sets up labeled backup LSP with the remote next-hop for the protected prefix. LDP also sets up another transport LSP to tunnel traffic to remote next-hop without exposing the LFA backup label as learnt from remote node.

Consider the topology shown in the figure below:

Figure 8: FRR with Remote LFA with Ring Topology



Device A tries to send traffic destined to F to next-hop B. Device B cannot be used as an LFA for prefixes advertised by nodes C and F. The actual LFA is node D. However, node D is not directly connected to the protecting node A. To protect prefixes advertised by C, node A must tunnel the packet around the failed link A-C to node D, provided that the tunnel does not traverse the failing link.

FRR Remote LFA feature enables you to tunnel a packet around a failed link to a remote loop-free alternate that is more than one hop away. In the figure above, the green arrow between A and D shows the tunnel that is automatically created by the remote LFA feature to bypass looping.

Configuration

Perform the following tasks to configure FRR with LFA.

Configure FRR with local LFA

```
/* Configure FRR with local LFA using IS-IS */
Router# configure
Router(config)# router isis ring
Router(config)#  is-type level-1
Router(config-isis)# net 49.0001.0000.0000.0007.00
Router(config-isis)# nsf cisco
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# mpls traffic-eng level-1-2
Router(config-isis-af)# mpls traffic-eng router-id 10.7.7.7
Router(config-isis-af)# exit
Router(config-isis)# interface Loopback 0
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if)# exit
Router(config-isis)# interface TenGigabitEthernet 0/0/0/4
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-af)# fast-reroute per-prefix
Router(config-isis-af)# commit

/* Configure FRR with local LFA using OSPF*/
Router# configure
Router(config)# router ospf 50
Router(config-ospf)# router-id 10.1.1.1
Router(config-ospf)# address-family ipv4 unicast
Router(config-ospf-af)# area 0
Router(config-ospf-af)# mpls traffic-eng
Router(config-ospf-af)# interface Loopback 0
Router(config-ospf-af)# exit
Router(config-ospf)# interface HundredGigE0/0/1/0
Router(config-ospf-if)# fast-reroute per-prefix
Router(config-ospf-if)# exit
Router(config-ospf)# exit
Router(config)# mpls traffic-eng router-id Loopback 0
```

Configure remote FRR with remote LFA.

```
/* Configure FRR with remote LFA using IS-IS */
Router# configure
Router(config)# router isis ring
Router(config)#  is-type level-1
Router(config-isis)# net 49.0001.0000.0000.0007.00
Router(config-isis)# nsf cisco
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# mpls traffic-eng level-1-2
```

```
Router(config-isis-af)# mpls traffic-eng router-id 10.7.7.7
Router(config-isis-af)# exit
Router(config-isis)# interface Loopback 0
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if)# exit
Router(config-isis)# interface TenGigabitEthernet 0/0/0/4
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-af)# fast-reroute per-prefix remote-lfa
Router(config-isis-af)# fast-reroute per-prefix remote-lfa prefix-list /* The prefix-list
option filters PQ node router ID based on prefix list */

Router(config-isis-af)# fast-reroute per-prefix remote-lfa tunnel mpls-ldp
Router(config-isis-af)# commit

/* Configure FRR with remote LFA using OSPF */
Router# configure
Router(config)# router ospf 50
Router(config-ospf)# router-id 10.1.1.1
Router(config-ospf)# address-family ipv4 unicast
Router(config-ospf-af)# area 0
Router(config-ospf-af)# mpls traffic-eng
Router(config-ospf-af)# interface Loopback 0
Router(config-ospf-af)# exit
Router(config-ospf)# interface HundredGigE0/0/1/0
Router(config-ospf-if)# fast-reroute per-prefix
Router(config-ospf-if)# fast-reroute per-prefix remote-lfa tunnel mpls-ldp
Router(config-ospf-if)# exit
Router(config-ospf)# exit
Router(config)# mpls traffic-eng router-id Loopback 0
```

Running Configuration

This section shows the FRR with local LFA configuration.

```
/* FRR with local LFA with ISIS */
router isis ring
 is-type level-1
 net 49.0001.0000.0000.0007.00
 nsf cisco
 address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-1-2
  mpls traffic-eng router-id 10.7.7.7
 !
 interface Loopback0
  point-to-point
  address-family ipv4 unicast
  !
 !
 interface HundredGigE0/0/1/0
  point-to-point
  address-family ipv4 unicast
fast-reroute per-prefix

/* FRR with local LFA with OSPF */
router ospf 50
 router-id 10.1.1.1
address-family ipv4 unicast
```

```
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface HundredGigE0/0/1/0
 fast-reroute per-prefix
  !
!
 mpls traffic-eng router-id loopback 0
!
```

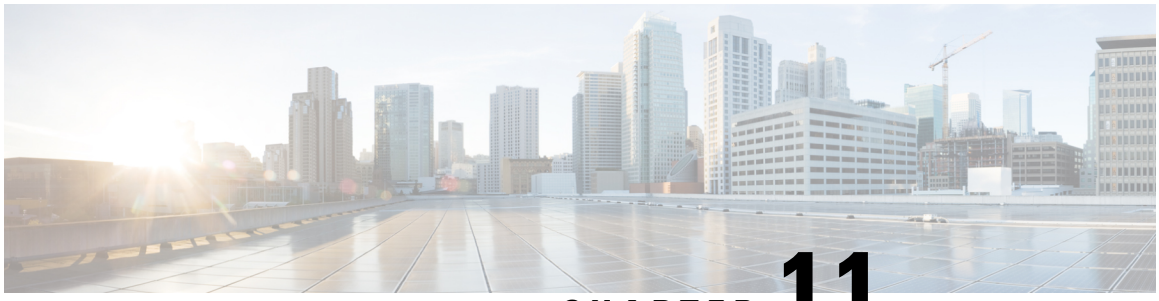
This section shows the FRR with remote LFA configuration.

```
/* FRR with remote  LFA with ISIS */
ipv4 prefix-list RLFA
 10 deny 3.3.3.3/32
 20 permit 0.0.0.0/0 le 32
router isis ring
 is-type level-1
 net 49.0001.0000.0000.0007.00
 nsf cisco
 address-family ipv4 unicast
 fast-reroute per-prefix remote-lfa prefix-list RLFA
  metric-style wide
  mpls traffic-eng level-1-2
  mpls traffic-eng router-id 10.7.7.7
 !
 interface Loopback0
  point-to-point
  address-family ipv4 unicast
  !
 !
 interface TenGigabitEthernet 0/0/0/4
  point-to-point
  address-family ipv4 unicast
fast-reroute per-prefix
fast-reroute per-prefix remote-lfa tunnel mpls-ldp


/* FRR with remote  LFA with OSPF */
router ospf 50
 router-id 10.1.1.1
address-family ipv4 unicast
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface HundredGigE0/0/1/0
 fast-reroute per-prefix
fast-reroute per-prefix remote-lfa tunnel mpls-ldp
  !
!
 mpls traffic-eng router-id loopback 0
!
```

Verification

The show outputs given in the following section display the details of the configuration of the FRR with Remote LFA feature, and the status of their configuration.

```
/* Verify the route summary information about the specified routing table. */

RP/0/RP0/CPU0:router# show route 10.3.3.3

Routing entry for 10.3.3.3/32
   Known via "isis 44", distance 115, metric 20, type level-1
   Installed Nov 15 19:43:13.367 for 00:00:34
   Routing Descriptor Blocks
     10.1.1.1, from 10.3.3.3, via TenGigE0/0/0/0, Backup (remote)
       Remote LFA is 10.9.9.9
      Route metric is 0
     10.1.1.2, from 10.3.3.3, via TenGigE0/7/0/3, Protected
       Route metric is 20
   No advertising protos.


/* Verify the MPLS LDP configuration. */

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

 Prefix           Label   Label(s)       Outgoing     Next Hop            Flags
                  In      Out            Interface                        G S R
 ---------------- ------ -------- --------- --------------- ----
 192.0.2.0/24     16019   { 16001        Te0/0/0/0    10.1.1.1        (!)     R
                            28006 }                  (10.9.9.9)
                  ImpNull        Te0/7/0/3    192.0.2.1
 192.0.2.1/32     16013   ImpNull        Te0/0/0/0    10.1.1.1
 192.0.1.0/32     16014   { 16001        Te0/0/0/0    10.1.1.1        (!)     R
                            16002 }                  (10.9.9.9)
                  ImpNull        Te0/7/0/3    192.0.2.2
 10.9.9.9/32      16012   16001          Te0/0/0/0    10.1.1.1
                  28006          Te0/7/0/3    192.0.2.1
 10.23.1.0/24     16018   16004          Te0/0/0/0    10.1.1.1        (!)
                  ImpNull        Te0/7/0/3    192.0.2.1
 10.34.1.0/24     16015   ImpNull        Te0/0/0/0    10.1.1.1
 10.0.0.1/32      16011  { 16001         Te0/0/0/0     10.1.1.1        (!)     R
                            16013 }                  (10.9.9.9)
                  16016          Te0/7/0/3    192.0.2.1
 10.100.0.2/32     16010   { 16001       Te0/0/0/0    10.1.1.1         (!)     R


/* Verify whether RLFA filtering is active */
RP/0/0/CPU0:Router #show isis fast-reroute 1.0.0.2/32 detail

L2 1.0.0.2/32 [20/115] medium priority
     via 1.2.0.2, GigabitEthernet0/0/0/0, R2, Weight: 0
       Backup path: R-LFA, via R3 [1.0.0.3], via 1.4.1.2, GigabitEthernet0/0/0/1 R4, Weight:
 0, Metric: 20 /*3.3.3.3 is filtered out, and another address is picked when RLFA filtering
 is active */
       P: No, TM: 20, LC: No, NP: No, D: No, SRLG: Yes
     src R2.00-00, 1.0.0.2
```



CHAPTER 11

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.

This module describes how to implement EIGRP on your Cisco IOS XR network.

- Restrictions for Implementing EIGRP , on page 337
- Information About Implementing EIGRP, on page 337

Restrictions for Implementing EIGRP

The following restrictions are employed when running EIGRP on this version of Cisco IOS XR software:

-
- The characters allowed for EIGRP process name are `@ . # : - _` only.
- Simple Network Management Protocol (SNMP) MIB is not supported.
- Interface static routes are not automatically redistributed into EIGRP, because there are no network commands.
- Metric configuration (either through the **default-metric** command or a route policy) is required for redistribution of connected and static routes.
- Auto summary is disabled by default.
- Stub leak maps are not supported.

Information About Implementing EIGRP

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

EIGRP Functional Overview

Enhanced Interior Gateway Routing Protocol (EIGRP) is an interior gateway protocol suited for many different topologies and media. EIGRP scales well and provides extremely quick convergence times with minimal network traffic.

EIGRP has very low usage of network resources during normal operation. Only hello packets are transmitted on a stable network. When a change in topology occurs, only the routing table changes are propagated and not the entire routing table. Propagation reduces the amount of load the routing protocol itself places on the network. EIGRP also provides rapid convergence times for changes in the network topology.

The distance information in EIGRP is represented as a composite of available bandwidth, delay, load utilization, and link reliability with improved convergence properties and operating efficiency. The fine-tuning of link characteristics achieves optimal paths.

The convergence technology that EIGRP uses is based on research conducted at SRI International and employs an algorithm referred to as the Diffusing Update Algorithm (DUAL). This algorithm guarantees loop-free operation at every instant throughout a route computation and allows all devices involved in a topology change to synchronize at the same time. Routers that are not affected by topology changes are not involved in recomputations. The convergence time with DUAL rivals that of any other existing routing protocol.

EIGRP 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.

Configure an Authentication Keychain

Configure an Authentication Keychain for an IPv4/IPv6 Interface on a Default VRF

```
Router# config
Router (config)# router eigrp 100
Router (config-eigrp)# address-family ipv4
Router (config-eigrp-af)# interface gigabitEthernet 0/1/5/0
Router (config-eigrp-af-if)# authentication keychain keychain1
Router (config-eigrp-af-if)# commit
```

Running Configuration

```
config
 router eigrp 100
  address-family ipv4
   interface gigabitEthernet 0/1/5/0
   authentication keychain keychain1
  commit
```

Configure an Authentication Keychain for an IPv4/IPv6 Interface on a Nondefault VRF

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

```
Router (config-eigrp)# vrf vrf1
Router (config-eigrp-vrf)# address-family ipv4
Router (config-eigrp-vrf-af)# interface gigabitEthernet 0/1/5/0
Router (config-eigrp-vrf-af-if)# authentication keychain keychain1
Router (config-eigrp-vrf-af-if)# commit
```

Running Configuration

```
config
 router eigrp 100
  vrf vrf1
   address-family ipv4
    interface gigabitEthernet 0/1/5/0
     authentication keychain keychain1
      commit
```

Enable EIGRP Routing

This task enables EIGRP routing and establishes an EIGRP routing process.

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

```
Router# configure
Router (config)# router eigrp 100
Router (config-eigrp)# address-family ipv4
Router (config-eigrp-af)# router-id 172.20.1.1
Router (config-eigrp-af)# default-metric 1000 100 250 100 1500
Router (config-eigrp-af)# distance 80 130
Router (config-eigrp-af)# interface GigabitEthernet 0/1/0/0
Router (config-eigrp-af-if)# hold-time 30
Router (config-eigrp-af-if)# bandwidth-percent 75
Router (config-eigrp-af-if)# commit
```

Running Configuration

This section shows EIGRP routing running configuration.

```
configure
 router eigrp 100
  address-family ipv4
   router-id 172.20.1.1
   default-metric 1000 100 250 100 1500
   distance 80 130
   interface GigabitEthernet 0/1/0/0
    hold-time 30
    bandwidth-percent 75
   !
  !
!
```

Monitor EIGRP Routing

```
Router# config
Router (config)# router eigrp 100
Router (config-eigrp)# address-family ipv4
```

```
Router (config-eigrp-af)# log-neighbor-changes
Router (config-eigrp-af)# log-neighbor-warnings
Router (config-eigrp-af)# commit
Router # clear eigrp 20 neighbors GigabitEthernet 0/1/0/0

Router # clear eigrp topology 10.1.0.0/8
Router # show eigrp vrf all accounting
Router # show eigrp interfaces detail
Router # show eigrp neighbors detail static
Router # show eigrp topology summary
Router # show eigrp traffic
```

Running Configuration

```
config
 router eigrp 100
  address-family ipv4
  log-neighbor-changes
  log-neighbor-warnings
```

Configure Route Summarization for an EIGRP Process

This task configures route summarization for an EIGRP process.

You can configure a summary aggregate address for a specified interface. If any more specific routes are in the routing table, EIGRP advertises the summary address from the interface with a metric equal to the minimum of all more specific routes.

Note You should not use the **summary-address** summarization command to generate the default route (0.0.0.0) from an interface. This command creates an EIGRP summary default route to the null 0 interface with an administrative distance of 5. The low administrative distance of this default route can cause this route to displace default routes learned from other neighbors from the routing table. If the default route learned from the neighbors is displaced by the summary default route or the summary route is the only default route present, all traffic destined for the default route does not leave the router; instead, this traffic is sent to the null 0 interface, where it is dropped.

The recommended way to send only the default route from a given interface is to use a **route-policy** command.

```
Router# configure
Router (config)# router eigrp 100
Router (config-eigrp)# address-family ipv4
Router (config-eigrp-af)# route-policy FILTER_DEFAULT out
Router (config-eigrp-af)# interface GigabitEthernet 0/1/0/0
Router (config-eigrp-af-if)# summary-address 192.168.0.0/16 95
Router (config-eigrp-af-if)# commit
```

Running Configuration

This section shows route summarization for an EIGRP process running configuration.

```
configure
 router eigrp 100
  address-family ipv4
```

```
   route-policy FILTER_DEFAULT out
   interface GigabitEthernet 0/1/0/0
    summary-address 192.168.0.0/16 95
    !
   !
!
```

Configure Stub Routing for an EIGRP Process

Perform the task to configure the distribution and remote routers to use an EIGRP process for stub routing.

Note EIGRP stub routing should be used only on remote routers. A stub router is defined as a router connected to the network core or distribution layer through which core transit traffic should not flow. A stub router should not have any EIGRP neighbors other than distribution routers. Ignoring this restriction causes undesirable behavior.

```
Router# configure
Router (config)# router eigrp 100
Router (config-eigrp)# address-family ipv4
Router (config-eigrp-af)# stub receive-only
Router (config-eigrp-af)# commit
```

Running Configuration

This section shows stub routing for an EIGRP process running configuration.

```
configure
 router eigrp 100
  address-family ipv4
   stub receive-only
   !
  !
!
```

Configure 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.

```
Router# configure
Router (config)# router eigrp 100
Router (config-eigrp)# vrf vrf_A
Router (config-eigrp-vrf)# address-family ipv4
Router (config-eigrp-vrf-af)# router-id 33
Router (config-eigrp-vrf-af)# autonomous-system 2
Router (config-eigrp-vrf-af)# redistribute bgp 100
Router (config-eigrp-vrf-af)# interface gigabitEthernet 0/1/5/0
Router (config-eigrp-vrf-af-if)# site-of-origin 3:4
Router (config-eigrp-vrf-af-if)# commit
```

Running Configuration

This section shows EIGRP as a PE-CE protocol running configuration.

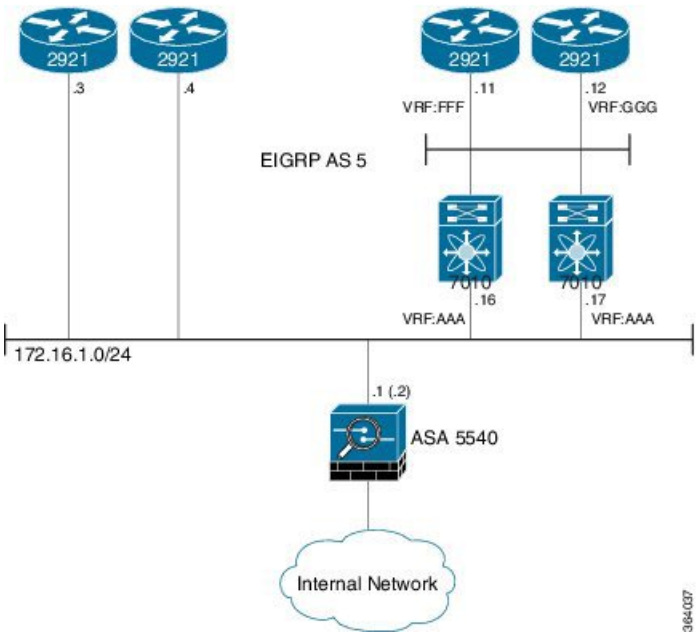
```
configure
 router eigrp 100
  vrf vrf_A
   address-family ipv4
    router-id 33
    autonomous-system 2
    redistribute bgp 100
    interface gigabitEthernet 0/1/5/0
     site-of-origin 3:4
    !
   !
!
```

Unicast Neighbors

EIGRP typically broadcasts or multicasts routing updates. For security reasons, you can opt to configure static neighbors in the EIGRP routing process, forcing EIGRP to communicate to specified neighbors using unicast. When you specify a static neighbor relationship over a particular interface, EIGRP disables the processing of multicast EIGRP packets on the specified interface. This ensures that EIGRP does not send nor process received multicast EIGRP traffic on an interface which has a static neighbor defined under the EIGRP routing process.

In cases where the neighbors are not adjacent, normal EIGRP peering mechanisms cannot be used to exchange EIGRP information. In order to support this type of network, EIGRP provides the neighbor command, which allows remote neighbors to be configured and sessions established though unicast packet transmission. However, as the number of forwarders needing to exchange EIGRP information over the networking cloud increases, unicast EIGRP neighbor definitions may become cumbersome to manage. Each neighbor must be manually configured, resulting in increased operational costs. To better accommodate deployment of these topologies, ease configuration management, and reduce operational costs, the Dynamic Neighbors feature provides support for the dynamic discovery of remote unicast (referred to as “remote neighbors”). Remote neighbor support allows EIGRP peering to one or more remote neighbors, which may not be known at the time the device is configured, thus reducing configuration management.

In the topology illustrated below, ASA behaves as a hub and the other routers (2921s, 7010s) act as spokes. The 2921's and 7010's must not peer with each other, and there must never be a time where a packet (data traffic) is routed in this path: ASA > 2921.3 > 2921.4. To support this type of network, EIGRP allows you to configure static neighbors and establish sessions using unicast packet transmission. Thus, in this topology, 2921s and 7010s peer with ASA using neighbor command and ASA is configured to dynamically discover remote neighbors.



Remote Neighbor Session Policy

When using remote unicast-listen or remote multicast-group neighbor configurations, EIGRP neighbor IP addresses are not predefined, and neighbors may be many hops away. A device with this configuration could peer with any device that sends a valid HELLO packet. Because of security considerations, this open aspect requires policy capabilities to limit peering to valid devices and to restrict the number of neighbors in order to limit resource consumption. This capability is accomplished using the following manually configured parameters, and takes effect immediately.

- **Neighbor Filter List**

  The optional allow-list keyword, available in the remote-neighbors command, enables you to use an access list (access control list) to specify the remote IP addresses from which EIGRP neighbor connections may be accepted. If you do not use the allow-list keyword, then all IP addresses (permit any) will be accepted. The access control list (ACL) defines a range of IPv4 or IPv6 IP addresses with the following conditions:

  - Any neighbor that has a source IP address that matches an IP address in the access list will be allowed (or denied) based on the user configuration.
  - If the allow-list keyword is not specified, any IP address will be permitted (permit any).
  - The allow-list keyword is supported only for remote multicast-group and unicast-listen neighbors. It is not available for static, remote static, or local neighbors.
  - Incoming EIGRP packets that do not match the specified access list will be rejected.

- **Maximum Remote Neighbors**

  The optional max-neighbors keyword, available in the remote-neighbors command, enables you to specify a maximum number of remote neighbors that EIGRP can create using the remote neighbor configurations. When the maximum number of remote neighbors has been created for a configuration, EIGRP rejects all subsequent connection attempts for that configuration. This option helps to protect against denial-of-service attacks that attempt to create many remote neighbors in an attempt to overwhelm device resources. The max-neighbors configuration option has the following conditions:

- This option is supported only for remote multicast-group or unicast-listen neighbors. It is not available for local, static, or remote static neighbors.
- There is no default maximum. If you do not specify a maximum number of remote neighbors, the number of remote neighbors is limited only by available memory and bandwidth.
- Reducing the maximum number of remote neighbors to less than the current number of sessions will result in the neighbors (in no specific order) being dropped until the count reaches the new limit.

- **Configuration Changes for the Neighbor Filter List and Maximum Number of Remote Neighbors**

  When the allow-list or max-neighbors configurations are changed, any existing remote EIGRP sessions that are no longer allowed by the new configuration will be removed automatically and immediately. Pre-existing neighbors that are still allowed by the new configuration will not be affected.

Understanding Neighbor Terms

The following terms are used when describing neighbor types:

- **local neighbor:** A neighbor that is adjacent on a shared subnet (or common subnet) and uses a link-local multicast address for packet exchange. This is the default type of neighbor in EIGRP.
- **static Neighbor:** Any neighbor that uses unicast to communicate, is one hop away, is on a common subnet, and whose IP address has been specified using the neighbor ip-address command.
- **remote neighbor:** Any neighbor that is multiple hops away, including Remote Static Neighbors.
- **remote group:** Any neighbor that is multiple hops away, does not have its address manually configured with the neighbor command and uses the multicast group address for packet exchange.
- **remote static neighbor:** Any neighbor that uses unicast to communicate, is multiple hops away, and whose IP address has been specified using the neighbor ip-address command.
- **remote unicast-listen (or simply unicast-listen):** Any neighbor that uses unicast to communicate, is multiple hops away, and whose IP address has not been configured using the neighbor ip-address command.
- **remote dnyamic:** Any neighbor that is multiple hops away and whose address has not been configured with the neighbor ip-address command, that is, a remote multicast-group or remote unicast-listen neighbor, but not a remote static neighbor.

Remote Unicast-Listen (Point-to-Point) Neighbors

For configurations in which multiple remote neighbors peer with a single hub (point-to-point), the hub can be configured for remote unicast-listen peering using the remote-neighbors command to allow the remote neighbors to peer with the hub without having to manually configure the remote neighbor IP addresses on the hub. When configured with this command, the hub device:

- Uses its interface IP address as the source IP address for any unicast transmissions. This IP address must be routable.
- Requires neighbors that peer with the hub to be configured using the neighbor ip-address loopback loopback-interface-number remote maximum-hops command where ip-address is the unicast address of the local device interface.
- Listens for unicast HELLO packets on the interface specified in the remote-neighbor command.
- Accepts a unicast HELLO packet if it is in the IP address range configured using the allow-list keyword, or any unicast HELLO packet if an allow list is not defined.
- Rejects multicast HELLO packets from any neighbor that is also sending unicast HELLO packets and is permitted by the unicast allow list (or all neighbors if an allow list is not defined).

- Begins normal neighbor establishment using the IP addresses of the remote neighbors for packet transmission once the neighbor relationship is established.
- When remote-neighbor command is configured on an interface, the router will only start sending HELLOs on that interface if it has atleast one neighbor and only to those neighbors from which it has received HELLOs.
- On an interface if dynamic neighbors already exist and remote-neighbor unicast- listen is configured, then the existing neighbor relationships will be torn down and only unicast-neighbor relationships will be allowed there after.

Restrictions for remote neighbors

A single unicast address can only be configured to a single remote static neighbor for a given address-family. You cannot configure a second remote static neighbor using the same unicast address, on a different interface. EIGRP configuration of remote neighbors under different address families is unrestricted.

A single interface can be configured under a single address family with a single unicast-listen remote-neighbors command and with any number of static and remote static neighbors (each using a different unicast address).

Inheritance and precedence of the remote neighbor configurations

Static neighbors configured with the neighbor <*address*> or neighbor <*address*> remote commands take precedence over the remote neighbors created as a result of the remote-neighbors command. If the remote address of an incoming unicast EIGRP connection matches both a static neighbor and the remote unicast-listen neighbor access list, the static neighbor is used and no remote unicast-listen neighbor is created. If you configure a new static neighbor while a remote neighbor for the same remote address already exists, EIGRP automatically removes the remote unicast-listen neighbor.

EIGRP Features

EIGRP offers the following features:

- Fast convergence—The DUAL algorithm allows routing information to converge as quickly as any currently available routing protocol.
- Partial updates—EIGRP sends incremental updates when the state of a destination changes, instead of sending the entire contents of the routing table. This feature minimizes the bandwidth required for EIGRP packets.
- Neighbor discovery mechanism—This is a simple hello mechanism used to learn about neighboring routers. It is protocol independent.
- Variable-length subnet masks (VLSMs).
- Arbitrary route summarization.
- Scaling—EIGRP scales to large networks.

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

- Support for IPv4 and IPv6 address families.
- Provider Edge (PE)-Customer Edge (CE) protocol support with Site of Origin (SoO) and Border Gateway Protocol (BGP) cost community support.
- PECE protocol support for MPLS.

EIGRP Components

EIGRP has the following four basic components:

- Neighbor discovery or neighbor recovery
- Reliable transport protocol
- DUAL finite state machine
- Protocol-dependent modules

Neighbor discovery or neighbor recovery is the process that routers use to dynamically learn of other routers on their directly attached networks. Routers must also discover when their neighbors become unreachable or inoperative. Neighbor discovery or neighbor recovery is achieved with low overhead by periodically sending small hello packets. As long as hello packets are received, the Cisco IOS XR software can determine that a neighbor is alive and functioning. After this status is determined, the neighboring routers can exchange routing information.

The reliable transport protocol is responsible for guaranteed, ordered delivery of EIGRP packets to all neighbors. It supports intermixed transmission of multicast and unicast packets. Some EIGRP packets must be sent reliably and others need not be. For efficiency, reliability is provided only when necessary. For example, on a multiaccess network that has multicast capabilities (such as Ethernet) it is not necessary to send hello packets reliably to all neighbors individually. Therefore, EIGRP sends a single multicast hello with an indication in the packet informing the receivers that the packet need not be acknowledged. Other types of packets (such as updates) require acknowledgment, which is indicated in the packet. The reliable transport has a provision to send multicast packets quickly when unacknowledged packets are pending. This provision helps to ensure that convergence time remains low in the presence of various speed links.

The DUAL finite state machine embodies the decision process for all route computations. It tracks all routes advertised by all neighbors. DUAL uses the distance information (known as a metric) to select efficient, loop-free paths. DUAL selects routes to be inserted into a routing table based on a calculation of the feasibility condition. A successor is a neighboring router used for packet forwarding that has a least-cost path to a destination that is guaranteed not to be part of a routing loop. When there are no feasible successors but there are neighbors advertising the destination, a recomputation must occur. This is the process whereby a new successor is determined. The amount of time required to recompute the route affects the convergence time. Recomputation is processor intensive; it is advantageous to avoid unneeded recomputation. When a topology change occurs, DUAL tests for feasible successors. If there are feasible successors, it uses any it finds to avoid unnecessary recomputation.

The protocol-dependent modules are responsible for network layer protocol-specific tasks. An example is the EIGRP module, which is responsible for sending and receiving EIGRP packets that are encapsulated in IP. It is also responsible for parsing EIGRP packets and informing DUAL of the new information received. EIGRP asks DUAL to make routing decisions, but the results are stored in the IP routing table. EIGRP is also responsible for redistributing routes learned by other IP routing protocols.

EIGRP Configuration Grouping

Cisco IOS XR Software groups all EIGRP configuration under router EIGRP configuration mode, including interface configuration portions associated with EIGRP. To display EIGRP configuration in its entirety, use the **show running-config router eigrp** command. The command output displays the running configuration for the configured EIGRP instance, including the interface assignments and interface attributes.

EIGRP Configuration Modes

The following examples show how to enter each of the configuration modes. From a mode, you can enter the **?** command to display the commands available in that mode.

Router Configuration Mode

The following example shows how to enter router configuration mode:

```
Router# configuration
Router(config)# router eigrp 100
Router(config-eigrp)#
```

VRF Configuration Mode

```
Router# configuration
  Router(config)# router eigrp 100
  Router(config-eigrp)# vrf customer1
  Router(config-eigrp-vrf)#
```

IPv4 Address Family Configuration Mode

The following example shows how to enter IPv4 address family configuration mode:

```
Router# configuration
  Router(config)# router eigrp 100
  Router(config-eigrp)# address-family ipv4
  Router(config-eigrp-af)#
```

IPv6 Address Family Configuration Mode

```
Router# configuration
Router(config)# router eigrp 100
Router(config-eigrp)# address-family ipv6
Router(config-eigrp-vrf-af)#
```

IPv4 VRF Address Family Configuration Mode

The following example shows how to enter IPv4 VRF address family configuration mode:

```
  Router# configuration
  Router(config)# router eigrp 100
  Router(config-eigrp)# vrf customer1
  Router(config-eigrp-vrf)# address-family ipv4
  Router(config-eigrp-vrf-af)#
```

IPv6 VRF Address Family Configuration Mode

The following example shows how to enter IPv6 VRF address family configuration mode:

```
Router# configuration
Router(config)# router eigrp 100
Router(config-eigrp)# vrf customer1
Router(config-eigrp-vrf)# address-family ipv6
Router(config-eigrp-vrf-af)#
```

Interface Configuration Mode

The following example shows how to enter interface configuration mode in IPv4 address family configuration mode:

```
Router# configuration
Router(config)# router eigrp 100
Router(config-eigrp)# address-family ipv4
Router(config-eigrp-af)# interface GigabitEthernet 0/3/0/0
Router(config-eigrp-af-if)#
```

The following example shows how to enter interface configuration mode in IPv6 VRF configuration mode:

```
Router# configuration
Router(config)# router eigrp 100
Router(config-eigrp)# vrf customer1
Router(config-eigrp-vrf)# address-family ipv6
Router(config-eigrp-vrf-af)# interface POS0/5/0/0
Router(config-eigrp-vrf-af-if)#
```

EIGRP Interfaces

EIGRP interfaces can be configured as either of the following types:

- Active—Advertises connected prefixes and forms adjacencies. This is the default type for interfaces.
- Passive—Advertises connected prefixes but does not form adjacencies. The **passive** command is used to configure interfaces as passive. Passive interfaces should be used sparingly for important prefixes, such as loopback addresses, that need to be injected into the EIGRP domain. If many connected prefixes need to be advertised, then the redistribution of connected routes with the appropriate policy should be used instead.

MPLS VPN Support for EIGRP Between PE and CE

The MPLS VPN Support for EIGRP Between PE and CE feature allows service providers to configure the Enhanced Interior Gateway Routing Protocol (EIGRP) between provider edge (PE) and customer edge (CE) devices in a Multiprotocol Label Switching (MPLS) virtual private network (VPN) and offer MPLS VPN services to those customers that require native support for EIGRP. An MPLS VPN consists of a set of sites that are interconnected by an MPLS provider core network. At each customer site, one or more CE devices attach to one or more PE devices.

Redistribution for an EIGRP Process

Routes from other protocols can be redistributed into EIGRP. A route policy can be configured along with the **redistribute** command. A metric is required, configured either through the **default-metric** command or under the route policy configured with the **redistribute** command to import routes into EIGRP.

A route policy allows the filtering of routes based on attributes such as the destination, origination protocol, route type, route tag, and so on. When redistribution is configured under a VRF, EIGRP retrieves extended communities attached to the route in the routing information base (RIB). The SoO is used to filter out routing loops in the presence of MPLS VPN backdoor links.

Redistribute Routes for EIGRP

This task explains how to redistribute routes and apply limits on the number of routes. .

```
Router# configure
Router (config)# router eigrp 100
Router (config-eigrp)# address-family ipv4
Router (config-eigrp-af)# redistribute bgp 100
Router (config-eigrp-af)# redistribute maximum-prefix 5000 95 warning-only
Router (config-eigrp-af)# maximum paths 10
Router (config-eigrp-af)# maximum-prefix 50000
Router (config-eigrp-af)# commit
```

Running Configuration

This section shows redistribute routes for EIGRP running configuration.

```
configure
 router eigrp 100
  address-family ipv4
   redistribute bgp 100
   redistribute maximum-prefix 5000 95 warning-only

   maximum paths 10
   maximum-prefix 50000
   !
  !
!
```

Create a Route Policy and Attach 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.

```
Router# configure
Router (config)# route-policy IN-IPv4
Router (config-rpl)# set eigrp-metric 42 100 200 100 1200
Router (config-rpl)# end-policy
Router (config)# commit
Router (config)# router eigrp 100
Router (config-eigrp)# address-family ipv4
Router (config-eigrp-af)# route-policy IN-IPv4 in
Router (config-eigrp-af-rpl)# commit
```

Running Configuration

This section shows the running configuration.

```
configure
 route-policy IN-IPv4
 set eigrp-metric 42 100 200 100 1200
 end-policy
!
```

```
 router eigrp 100
  address-family ipv4
   route-policy IN-IPv4 in
   !
  !
!
```

Redistribute 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.

```
Router# configure
Router (config)# router eigrp 100
Router (config-eigrp)# vrf vrf_A
Router (config-eigrp-vrf)# address-family ipv4
Router (config-eigrp-vrf-af)# redistribute bgp 100
Router (config-eigrp-vrf-af)# route-policy policy_A in
Router (config-eigrp-vrf-af)# default-metric 1000 100 250 100 1500
Router (config-eigrp-vrf-af)# commit
```

Running Configuration

This section shows redistribute BGP routes into EIGRP running configuration.

```
configure
 router eigrp 100
  vrf vrf_A
   address-family ipv4
    redistribute bgp 100
    route-policy policy_A in
    default-metric 1000 100 250 100 1500
    !
   !
!
```

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:

```
Router/CPU0:Mar 13 08:19:55:eigrp[163]:%ROUTING-EIGRP-5-NBRCHANGE:IP-EIGRP(0) 1:Neighbor
11.0.0.20 (GigabitEthernet0/6/0/0) is down: K-value mismatch
```

Two scenarios occur in which this error message can be displayed:

- The two routers are connected on the same link and configured to establish a neighbor relationship. However, each router is configured with different K values.

  The following configuration is applied to ROUTER-A. The K values are changed with the **metric weights** command. A value of 2 is entered for the *k1* argument to adjust the bandwidth calculation. The value of 1 is entered for the *k3* argument to adjust the delay calculation.

  ```
  hostname ROUTER-A!
  interface GigabitEthernet0/6/0/0
   ipv4 address 10.1.1.1 255.255.255.0

  router eigrp 100
   metric weights 0 2 0 1 0 0
   interface GigabitEthernet0/6/0/0
  ```

  The following configuration is applied to ROUTER-B. However, the **metric weights** command is not applied and the default K values are used. The default K values are 1, 0, 1, 0, and 0.

  ```
  hostname ROUTER-B!
  interface GigabitEthernet0/6/0/1
   ipv4 address 10.1.1.2 255.255.255.0

  router eigrp 100
   interface GigabitEthernet0/6/0/1
  ```

  The bandwidth calculation is set to 2 on ROUTER-A and set to 1 (by default) on ROUTER-B. This configuration prevents these peers from forming a neighbor relationship.

- The K-value mismatch error message can also be displayed if one of the two peers has transmitted a “goodbye” message and the receiving router does not support this message. In this case, the receiving router interprets this message as a K-value mismatch.

Goodbye Message

The goodbye message is a feature designed to improve EIGRP network convergence. The goodbye message is broadcast when an EIGRP routing process is shut down to inform adjacent peers about the impending

topology change. This feature allows supporting EIGRP peers to synchronize and recalculate neighbor relationships more efficiently than would occur if the peers discovered the topology change after the hold timer expired.

The following message is displayed by routers that run a supported release when a goodbye message is received:

```
Router/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:

```
Router/CPU0:Mar 13 09:13:17:eigrp[163]:%ROUTING-EIGRP-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor
 10.0.0.20 (GigabitEthernet0/6/0/0) is down: K-value mismatch
```

Note The receipt of a goodbye message by a nonsupporting peer does not disrupt normal network operation. The nonsupporting peer terminates the session when the hold timer expires. The sending and receiving routers reconverge normally after the sender reloads.

Percentage of Link Bandwidth Used for EIGRP Packets

By default, EIGRP packets consume a maximum of 50 percent of the link bandwidth, as configured with the **bandwidth** interface configuration command. You might want to change that value if a different level of link utilization is required or if the configured bandwidth does not match the actual link bandwidth (it may have been configured to influence route metric calculations).

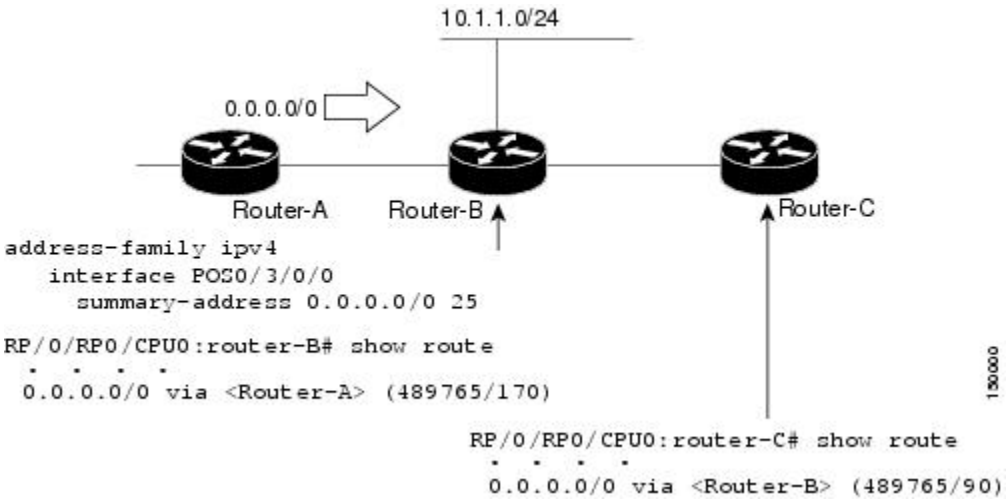
Floating Summary Routes for an EIGRP Process

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

The figure *Floating Summary Route is Applied to Router-B* 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:

```
Router(config)# router eigrp 100
Router(config-eigrp)# address-family ipv4
Router(config-eigrp-af)# interface GigabitEthernet 0/3/0/0
Router(config-eigrp-af-if)# summary-address 100.0.0.0 0.0.0.0
```

Figure 9: Floating Summary Route Is Applied to Router-B



The configuration of the default summary route on Router-B sends a 0.0.0.0/0 summary route to Router-C and blocks all other routes, including the 10.1.1.0/24 route, from being advertised to Router-C. However, this configuration also generates a local discard route on Router-B, a route for 0.0.0.0/0 to the null 0 interface with an administrative distance of 5. When this route is created, it overrides the EIGRP learned default route. Router-B is no longer able to reach destinations that it would normally reach through the 0.0.0.0/0 route.

This problem is resolved by applying a floating summary route to the interface on Router-B that connects to Router-C. The floating summary route is applied by relating an administrative distance to the default summary route on the interface of Router-B with the following statement:

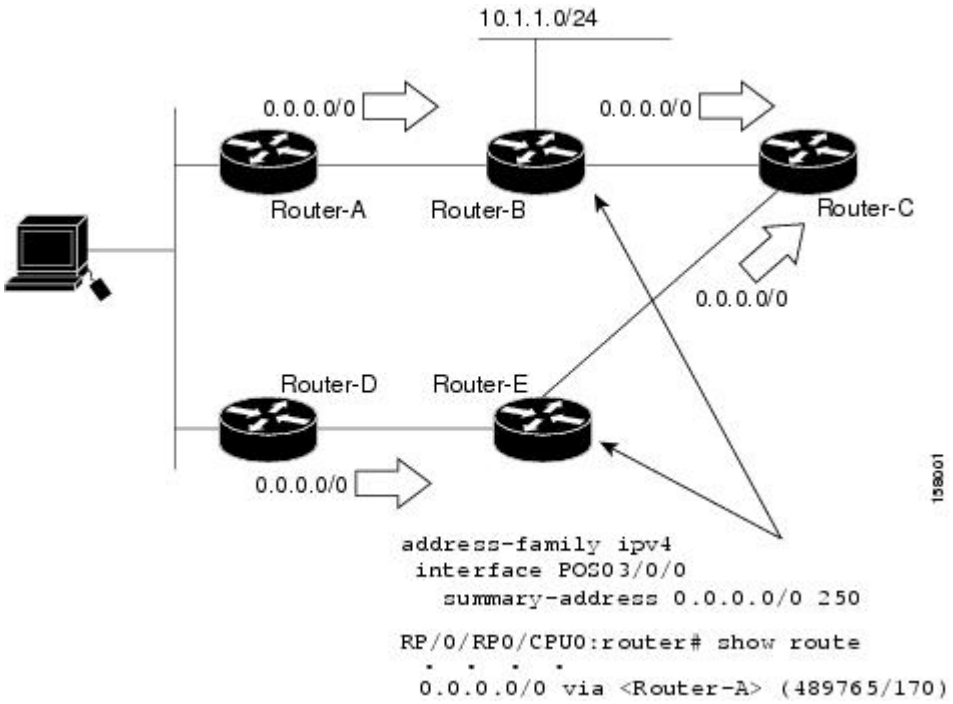
```
Router(config-if)# summary-address 100 0.0.0.0 0.0.0.0 250
```

The administrative distance of 250, applied in the above statement, is now assigned to the discard route generated on Router-B. The 0.0.0.0/0, from Router-A, is learned through EIGRP and installed in the local routing table. Routing to Router-C is restored.

If Router-A loses the connection to Router-B, Router-B continues to advertise a default route to Router-C, which allows traffic to continue to reach destinations attached to Router-B. However, traffic destined for networks to Router-A or behind Router-A is dropped when the traffic reaches Router-B.

The figure *Floating Summary Route is Applied for Dual-Homed Remotes* 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 10: Floating Summary Route Applied for Dual-Homed Remotes



However, if the link between Router-D and Router-E fails, the network may dump traffic into a black hole because Router-E continues to advertise the default route (0.0.0.0/0) to Router-C, as long as at least one link (other than the link to Router-C) to Router-E is still active. In this scenario, Router-C still forwards traffic to Router-E, but Router-E drops the traffic creating the black hole. To avoid this problem, you should configure the summary address with an administrative distance on only single-homed remote routers or areas in which only one exit point exists between the segments of the network. If two or more exit points exist (from one segment of the network to another), configuring the floating default route can cause a black hole to form.

Split Horizon for an EIGRP Process

Split horizon controls the sending of EIGRP update and query packets. When split horizon is enabled on an interface, update and query packets are not sent for destinations for which this interface is the next hop. Controlling update and query packets in this manner reduces the possibility of routing loops.

By default, split horizon is enabled on all interfaces.

Split horizon blocks route information from being advertised by a router on any interface from which that information originated. This behavior usually optimizes communications among multiple routing devices, particularly when links are broken. However, with nonbroadcast networks (such as Frame Relay and SMDS), situations can arise for which this behavior is less than ideal. For these situations, including networks in which you have EIGRP configured, you may want to disable split horizon.

Adjustment of Hello Interval and Hold Time for an EIGRP Process

You can adjust the interval between hello packets and the hold time.

Routing devices periodically send hello packets to each other to dynamically learn of other routers on their directly attached networks. This information is used to discover neighbors and learn when neighbors become unreachable or inoperative. By default, hello packets are sent every 5 seconds.

You can configure the hold time on a specified interface for a particular EIGRP routing process designated by the autonomous system number. The hold time is advertised in hello packets and indicates to neighbors the length of time they should consider the sender valid. The default hold time is three times the hello interval, or 15 seconds.

Stub Routing for an EIGRP Process

The EIGRP Stub Routing feature improves network stability, reduces resource usage, and simplifies stub router configuration.

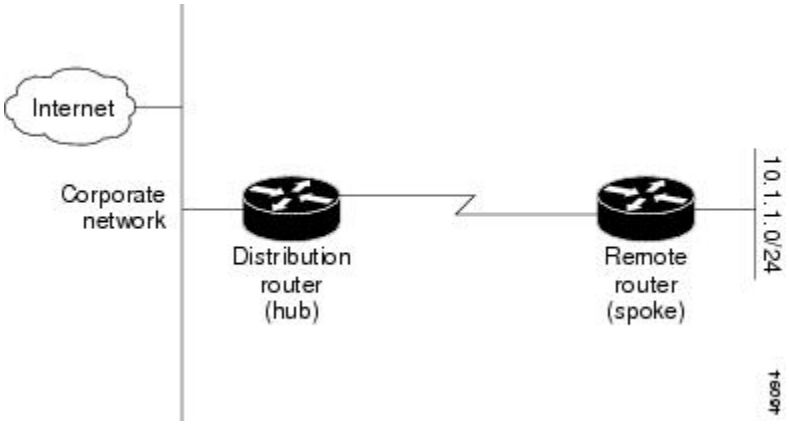
Stub routing is commonly used in a hub-and-spoke network topology. In a hub-and-spoke network, one or more end (stub) networks are connected to a remote router (the spoke) that is connected to one or more distribution routers (the hub). The remote router is adjacent only to one or more distribution routers. The only route for IP traffic to follow into the remote router is through a distribution router. This type of configuration is commonly used in WAN topologies in which the distribution router is directly connected to a WAN. The distribution router can be connected to many more remote routers. Often, the distribution router is connected to 100 or more remote routers. In a hub-and-spoke topology, the remote router must forward all nonlocal traffic to a distribution router, so it becomes unnecessary for the remote router to hold a complete routing table. Generally, the distribution router need not send anything more than a default route to the remote router.

When using the EIGRP Stub Routing feature, you need to configure the distribution and remote routers to use EIGRP and configure only the remote router as a stub. Only specified routes are propagated from the remote (stub) router. The stub router responds to all queries for summaries, connected routes, redistributed static routes, external routes, and internal routes with the message “inaccessible.” A router that is configured as a stub sends a special peer information packet to all neighboring routers to report its status as a stub router.

Any neighbor that receives a packet informing it of the stub status does not query the stub router for any routes, and a router that has a stub peer does not query that peer. The stub router depends on the distribution router to send the proper updates to all peers.

Figure 11: Simple Hub-and-Spoke Network

This figure shows a simple hub-and-spoke configuration.



The stub routing feature by itself does not prevent routes from being advertised to the remote router. In the example in the figure Simple Hub-and-Spoke Network, 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:

```
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:

```
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.

Default-Accept-In

The default-accept-in attach point allows you to set and reset the conditional default flag for EIGRP routes by evaluating the attached policy.

The following example shows a policy that sets the conditional default flag for all routes that match 10.0.0.0/8 and longer prefixes up to 10.0.0.0/25:

```
route-policy eigrp-cd-policy-in
  if destination in (10.0.0.0/8 ge 8 le 25) then
    pass
  endif
end-policy
!
router eigrp 100
  address-family ipv4
    default-information allowed in route-policy eigrp-cd-policy-in
    .
    .
    .
```

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.

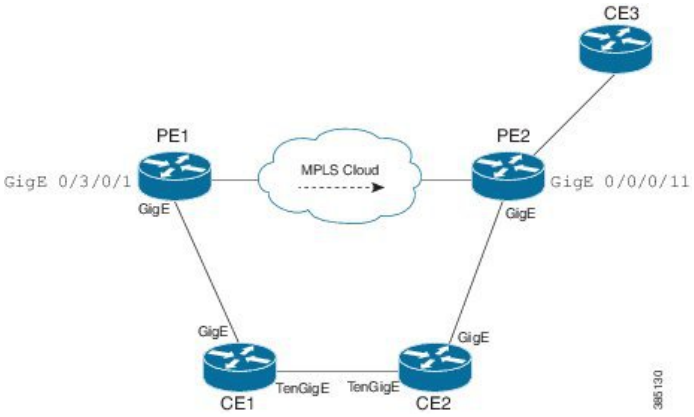
Route Manipulation using SoO match condition

The SoO configuration in EIGRP network can be used to manipulate routes using the SoO match condition in the routing policy. The egress interface of a PE router is used to compare and manipulate routes based on the SoO configuration on the remote PE router.

Topology

In the following topology, CE1, CE2 and CE3 are the customer edge routers. PE1 and PE2 are the provider edge routers. By default, CE1 will use PE1->PE2 to reach CE3.To configure CE1 to use CE2 to reach CE3, the metric advertised by PE1 must be increased.

The routing policy on PE1 manipulates routes received from CE3 via PE2, by using the SoO match condition. With this feature added, PE1 can increase the metric while advertising routes to CE1.



Configuration:

```
/*SoO tag is assigned on PE2 router*/

router(config)#interface GigabitEthernet0/0/0/11
router (config-if)#site-of-origin 33.33.33.33:33

/* A route-policy defined on PE1 */

router(config)#route-policy test
router(config-rpl)#if extcommunity soo matches-any (33.33.33.33:33) then
router(config-rpl-if)#set eigrp-metric 2121212121 333333333 245 250 1455
router(config-rpl-if)#endif
router(config-rpl)#end-policy
router (config)#commit
router (config)#

router(config)#interface GigabitEthernet0/3/0/1
```

```
router (config-if)#route-policy test out
```

Verification:

```
/*A route with poor metric advertised by PE1 is installed into CE1's routing table for SoO
 of site C3. */

router#show eigrp topology 6:6::1/128

IPv6-EIGRP AS(100): Topology entry for 6:6::1/128
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 15539149614794, RIB is
4294967295 Routing Descriptor Blocks: fe80::226:98ff:fe24:5109 (GigabitEthernet0/0/0/15),
from fe80::226:98ff:fe24:5109, Send flag is 0x0
Composite metric is (15539149614794/15539148304382), Route is Internal Vector metric:
Minimum bandwidth is 1000000 Kbit
Total delay is 237108596182784 picoseconds
Reliability is 245/255
Load is 250/255
Minimum MTU is 1455
Hop count is 2
Originating router is 2.2.2.2
Extended Community:
SoO:33.33.33.33:33
```

Note:

This feature is applicable to both ipv4 as well as ipv6.

All types of SoO(IP-Address, ASN2, ASN4) are supported.

EIGRP Wide Metric Computation

The Cisco IOS XR Enhanced Interior Gateway Routing Protocol (EIGRP) implementation is enhanced to perform wide metric computation. This enhancement is to support high bandwidth interfaces.

A new EIGRP command is added and existing EIGRP commands are enhanced to support wide metric computation feature.

- **metric rib-scale**—This command was introduced.
- **metric**—The **picoseconds** keyword was added.
- **metric weights**—Support was added for the *k6* constant.
- **show eigrp interfaces**—The command output was modified to display relevant wide metric information.
- **show eigrp neighbors** —The command output was modified to display relevant wide metric information.
- **show eigrp topology**—The command output was modified to display relevant wide metric information.
- **show protocols**—The command output was modified to display relevant wide metric information.

Note If there is a combination of IOS and IOS-XR PE devices in the network, then the EIGRP wide metric must be disabled in IOS-XR PE device. This is because the method of calculating metrics in L3VPN design between IOS and IOS-XR.

EIGRP Multi-Instance

The Enhanced Interior Gateway Routing Protocol (EIGRP) Multi-Instance feature allows multiple process instances to handle different routing instances and service the same VRF. Each process instance handles the routing instances configured under it. The multiple EIGRP process instance implementation allows to configure the EIGRP using a virtual-name in addition to an autonomous-system number.