CONTENTS

PREFACE

Preface xi
  Changes to This Document xi
  Communications, Services, and Additional Information xi

CHAPTER 1

New and Changed Routing Features 1
  New and Changed Routing Features 1

CHAPTER 2

Implementing IS-IS 3
  Enable IS-IS and Configure Level 1 or Level 2 Routing 3
  Single-Topology IPv6 5
    Configure Single Topology for IS-IS 5
    Set SPF Interval for a Single-Topology Configuration 10
  Customize Routes for IS-IS 12
  Set Priority for Adding Prefixes to RIB 15
  IS-IS Interfaces 16
    Tag IS-IS Interface Routes 16
  Limit LSP Flooding 19
    Control LSP Flooding for IS-IS 19
  IS-IS Authentication 23
    Configure Authentication for IS-IS 23
    Configure Keychains for IS-IS 25
  Nonstop Forwarding 26
    Configure Nonstop Forwarding for IS-IS 27
  ISIS NSR 29
    Configuring ISIS-NSR 29
    Configuring IS-IS Adjacency Stagger 31
CHAPTER 3

Implementing OSPF 47

Prerequisites for Implementing OSPF 48
Enable OSPF 48
Verify OSPF Configuration and Operation 50
Stub Area 53
Not-so-Stubby Area 53
Configure Stub and Not-So-Stubby Area Types 53
Neighbors and Adjacency for OSPF 56
Configure Neighbors for Nonbroadcast Networks 56
Authentication Strategies 60
  Configure Authentication at Different Hierarchical Levels for OSPF Version 2 60
Control Frequency That Same LSA Is Originated or Accepted for OSPF 63
Virtual Link and Transit Area for OSPF 65
  Create Virtual Link 66
Summarize Subnetwork LSAs on OSPF ABR 70
Route Redistribution for OSPF 73
  Redistribute Routes into OSPF 73
Nonstop Forwarding for OSPF Version 2 75
  Configure Nonstop Forwarding Specific to Cisco for OSPF Version 2 76
OSPF Shortest Path First Throttling 78
  Configure OSPF Shortest Path First Throttling 79
Graceful Restart for OSPFv3 81
  Configure OSPFv3 Graceful Restart 81
  Display Information About Graceful Restart 83
Warm Standby and Nonstop Routing for OSPF Version 2 83
  Enable Nonstop Routing for OSPFv2 84
Warm Standby and Nonstop Routing for OSPF Version 3 84
  Enable Nonstop Routing for OSPFv3 85
OSPFv2OSPF SPF Prefix Prioritization 85
  Configure OSPFv2 OSPF SPF Prefix Prioritization 87
Configure OSPF as a Provider Edge to Customer Edge (PE-CE) Protocol 90
Create Multiple OSPF Instances (OSPF Process and a VRF) 92
Label Distribution Protocol IGP Auto-configuration for OSPF 94
  Configure Label Distribution Protocol IGP Auto-configuration for OSPF 94
  Configure LDP IGP Synchronization: OSPF 95
OSPF Authentication Message Digest Management 97
  Configure Authentication Message Digest Management for OSPF 97
GTSM TTL Security Mechanism for OSPF 100
  Configure Generalized TTL Security Mechanism (GTSM) for OSPF 101
References for OSPF 103
  OSPF Functional Overview 103
  Comparison of Cisco IOS XR Software OSPFv3 and OSPFv2 104
  OSPF Hierarchical CLI and CLI Inheritance 105
Chapter 4: Implementing and Monitoring RIB

Implementing and Monitoring RIB 117

Verify RIB Configuration Using Routing Table 117
Verify Networking and Routing Problems 118
Disable RIB Next-hop Dampening 120
Enable RCC and LCC On-demand Scan 121
Enable RCC and LCC Background Scan 122
References for RIB 124

RIB Data Structures in BGP and Other Protocols 124
CHAPTER 5

Implementing Routing Policy 129

Restrictions for Implementing Routing Policy 129
Define Route Policy 130
Attach Routing Policy to BGP Neighbor 131
Modify Routing Policy Using Text Editor 132
References for Routing Policy 135

Routing Policy Language 136
Routing Policy Language Overview 136
Routing Policy Language Structure 136
Routing Policy Language Components 143
Routing Policy Language Usage 144

Policy Definitions 146

Parameterization 147
Parameterization at Attach Points 148
Global Parameterization 148

Semantics of Policy Application 149
Boolean Operator Precedence 149
Multiple Modifications of Same Attribute 149
When Attributes Are Modified 150
Default Drop Disposition 151
Control Flow 151
Policy Verification 152

Policy Statements 154
Remark 154
Disposition 154
Action 156
If 156
Boolean Conditions 157
apply 159
CHAPTER 6 Implementing Static Routes 193

Restrictions for Implementing Static Routes 193

Configure Static Route 194

Floating Static Routes 195

Configure Floating Static Route 195

Configure Static Routes Between PE-CE Routers 197

IPv4 Multicast Static Routes 198

Configure Multicast Static Routes 199

Default VRF 200

Associate VRF with a Static Route 200

Configure Native UCMP for Static Routing 201

References for Static Routes 202

Static Route Functional Overview 203

Default Administrative Distance 203

Directly Connected Routes 203

Floating Static Routes 204

Fully Specified Static Routes 204

Recursive Static Routes 204
## CHAPTER 7  Implementing BFD  207

BFD Overview  207

- Enabling BFD on a Static Route  208
- Enabling BFD for OSPF on an Interface  208
- Enable BFD for IS-IS on an Interface  209
- Enabling BFD on a BGP Neighbor  210
- Clear and Display BFD Counters  212

BFD over Bundle  212

- Enabling BFD Sessions on Bundle Members  212
- Specifying the BFD Destination Address on a Bundle  213
- Configuring the Minimum Thresholds for Maintaining an Active Bundle  214
- Configuring BFD Packet Transmission Intervals and Failure Detection Times on a Bundle  215
- Configure BFD over Bundles IETF Mode Support on a Per Bundle Basis  216

BFD Transparency  216

- Ethernet VPN Virtual Private Wire Service  217
  - Configuration  217
    - Running Configuration  218
    - Verification  219
- Bidirectional Forwarding Detection over Logical Bundle  221
  - Configuration  222
    - Running Configuration  223
    - Verification  224
- IPv4 Multihop BFD  226
  - Configure IPv4 Multihop BFD  226
  - Verification  227

## CHAPTER 8  Implementing UCMP  229

ECMP vs. UCMP Load Balancing  230

UCMP Minimum Integer Ratio  230

Configuring IS-IS With Weight  231

Configuring IS-IS With Metric  232

Configuring BGP With Weights  233

Configuring TE Tunnel With Weights  234
CHAPTER 9

Implementing Fast Reroute Loop-Free Alternate 249
  Prerequisites for Fast Reroute with Loop-Free Alternate 249
  Restrictions for Fast Reroute with Loop-Free Alternate 249
  IS-IS and FRR 250
  Repair Paths 250
  LFA Overview 250
  LFA Calculation 251
  Interaction Between RIB and Routing Protocols 251
  Fast Reroute with Remote Loop-Free Alternate 252
  Configuration 253
    Running Configuration 254
    Verification 255
Preface

The Routing Configuration Guide for Cisco NCS 5500 Series Routers preface contains these sections:

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

Changes to This Document

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

Table 1: Changes to This Document

<table>
<thead>
<tr>
<th>Date</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 2017</td>
<td>Initial release of this document.</td>
</tr>
<tr>
<td>March 2018</td>
<td>Republished for Release 6.3.2.</td>
</tr>
</tbody>
</table>

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you’re looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.
New and Changed Routing Features

This table summarizes the new and changed feature information for the Routing Configuration Guide for Cisco NCS 5500 Series Routers, and tells you where they are documented.

- New and Changed Routing Features, on page 1

### Table 2: Routing Features Added or Modified in IOS XR Release 6.3.x

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Changed in Release</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4 Multicast Static Routes</td>
<td>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.</td>
<td>Release 6.3.1</td>
<td>Implementing Static Routes, on page 193</td>
</tr>
</tbody>
</table>
### New and Changed Routing Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Changed in Release</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Fast Reroute with Remote Loop Free Alternate</td>
<td>Fast Reroute with Remote Loop-Free Alternate enables you to tunnel a packet around a failed link to a remote loop-free alternate that is more than one hop away.</td>
<td>Release 6.3.2</td>
<td>Fast Reroute with Remote Loop-Free Alternate, on page 252</td>
</tr>
<tr>
<td>UCMP Minimum Integer Ratio</td>
<td>This feature was introduced.</td>
<td>Release 6.3.2</td>
<td>Implementing UCMP chapter</td>
</tr>
<tr>
<td>IPv4 Multihop BFD</td>
<td>This feature was introduced.</td>
<td>Release 6.3.2</td>
<td>IPv4 Multihop BFD, on page 226</td>
</tr>
</tbody>
</table>
Implementing IS-IS


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 3
- Single-Topology IPv6, on page 5
- Customize Routes for IS-IS, on page 12
- Set Priority for Adding Prefixes to RIB, on page 15
- IS-IS Interfaces, on page 16
- Limit LSP Flooding, on page 19
- IS-IS Authentication, on page 23
- Nonstop Forwarding, on page 26
- ISIS NSR, on page 29
- Configuring IS-IS Adjacency Stagger, on page 31
- Multiprotocol Label Switching Traffic Engineering, on page 32
- IS-IS Overload Bit Avoidance, on page 38
- Configuring Global Weighted SRLG Protection, on page 39
- References for IS-IS, on page 41

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.
Before you begin

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

SUMMARY STEPS

1. configure
2. router isis instance-id
3. net network-entity-title
4. is-type { level-1 | level-1-2 | level-2-only }
5. commit
6. show isis [ instance instance-id ] protocol

DETAILED STEPS

Step 1  configure
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** commit  
**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

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 \texttt{adjacency-check disable} command, then an interface is not required to have the address family enabled.

2. The \texttt{single-topology} command is not valid in the \texttt{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 \texttt{address-family ipv4} stanza. You may configure the metric style in the \texttt{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 \texttt{address-family ipv6} stanza.

**SUMMARY STEPS**

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

Step 1
configure

Step 2
interface  type  interface-path-id

Example:

```
RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/1/0/3
```
Enters interface configuration mode.

Step 3
Do one of the following:

- ipv4 address  address mask
- ipv6 address  ipv6-prefix / prefix-length  [ eui-64 ]
- ipv6 address  ipv6-address  { / prefix-length | link-local }
- ipv6 enable

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.0.1.3 255.255.255.0
```
or
```
RP/0/RP0/CPU0:router(config-if)# ipv6 address 3ffe:1234:c18:1::/64 eui-64
RP/0/RP0/CPU0:router(config-if)# ipv6 address FE80::260:3EFF:FE11:6770 link-local
RP/0/RP0/CPU0:router(config-if)# ipv6 enable
```

or

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

or

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

or

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

or

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

- The link-local address can be used only to communicate with nodes on the same link.

- Specifying the ipv6 address ipv6-prefix / prefix-length interface configuration command without the eui-64 keyword configures site-local and global IPv6 addresses.

- Specifying the ipv6 address ipv6-prefix / prefix-length command with the eui-64 keyword configures site-local and global IPv6 addresses with an interface ID in the low-order 64 bits of the IPv6 address. Only the 64-bit network prefix for the address needs to be specified; the last 64 bits are automatically computed from the interface ID.

- Specifying the ipv6 address command with the link-local keyword configures a link-local address on the interface that is used instead of the link-local 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  

double-topology

Example:

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

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

- The double-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:
R/P/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  

Example:

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

Enters interface configuration mode.

**Step 11**

circuit-type  

Example:

R/P/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  

Example:

R/P/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**

**Step 14**

show isis  

Example:

R/P/0/RP0/CPU0:router# show isis interface HundredGigE 0/1/0/1

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

**Step 15**

show isis topology  

Example:

R/P/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.

```plaintext
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.

The incremental SPF algorithm can be enabled separately. When enabled, the incremental shortest path first (ISPF) is not employed immediately. Instead, the full SPF algorithm is used to “seed” the state information required for the ISPF to run. The startup delay prevents the ISPF from running for a specified interval after an IS-IS restart (to permit the database to stabilize). After the startup delay elapses, the ISPF is principally responsible for performing all of the SPF calculations. The reseed interval enables a periodic running of the full SPF to ensure that the ISFP state remains synchronized.

**SUMMARY STEPS**

1. configure
2. router isis `instance-id`
3. address-family `{ipv4 | ipv6}` `{unicast}`
4. spf-interval `{{initial-wait initial | secondary-wait secondary | maximum-wait maximum}}` `{level {1 | 2}}`
5. ispf `{level {1 | 2}}`
6. commit
## DETAILED STEPS

### Step 1
configure

### Step 2
router isis  instance-id

**Example:**

```bash
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
domain

**Example:**

```bash
RP/0/RP0/CPU0:router(config-router)# domain domain-id
```

Sets the IS-IS domain ID.

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

**Example:**

```bash
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
spf-interval  { [ initial-wait initial | secondary-wait secondary | maximum-wait maximum ] ... } [ level { 1 | 2 } ]

**Example:**

```bash
RP/0/RP0/CPU0:router(config-isis-af)# spf-interval initial-wait 10 maximum-wait 30
```

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

- This value imposes a delay in the SPF computation after an event trigger and enforces a minimum elapsed time between SPF runs.
- If this value is configured too low, the router can lose too many CPU resources when the network is unstable.
- Configuring the value too high delays changes in the network topology that result in lost packets.
- 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:**

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

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

### Step 6
commit

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

**Example:**

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

Displays IS-IS routing information.
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.

SUMMARY STEPS

1. configure
2. router isis instance-id
3. set-overload-bit [ on-startup { delay | wait-for-bgp } ] [ level { 1 | 2 } ]
4. address-family { ipv4 | ipv6 } [ unicast ]
5. default-information originate [ route-policy route-policy-name ]
6. distribute-list { { prefix-list prefix-list-name | route-policy route-policy-name } } in
7. redistribute isis instance [ level-1 | level-2 | level-1-2 ] [ metric metric ] [ metric-type { internal | external } ] [ policy policy-name ]
8. Do one of the following:
   - summary-prefix address / prefix-length [ level { 1 | 2 } ]
   - summary-prefix ipv6-prefix / prefix-length [ level { 1 | 2 } ]
9. maximum-paths route-number
10. distance weight [ address / prefix-length [ route-list-name ] ]
11. set-attached-bit
12. commit

DETAILED STEPS

Step 1 configure
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  commit

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 0/3/0/0
  address-family ipv4 unicast
!
router isis 2
  is-type level-1
  net 49.0002.0001.0001.0002.00
  address-family ipv4 unicast
  set -attached
  -bit
!
interface HundredGigE 0/1/0/0
  address-family ipv4 unicast
```

**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.

**SUMMARY STEPS**

1. `configure`
2. `router isis instance-id`
3. `address-family { ipv4 | ipv6 } [ unicast ]`
4. `metric-style wide [ transition ] [ level { 1 | 2 } ]`
5. `spf prefix-priority [ level { 1 | 2 } ] [ critical | high | medium ] { access-list-name | tag tag }`
6. `commit`

**DETAILED STEPS**

**Step 1**  
`configure`

**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**

```
commit
```

---

**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.
SUMMARY STEPS

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

DETAILED STEPS

Step 1 configure
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 0/1/0/3
```
Enters interface configuration mode.

**Step 7**

```
address-family { ipv4 | ipv6 } [ unicast ]
```

**Example:**

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

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

**Step 8**

```
tag  tag
```

**Example:**

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

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

**Step 9**

```
commit
```

**Step 10**

```
show isis [ ipv4 | ipv6 | afi-all ] [ unicast | safi-all ] route [ detail ]
```

**Example:**

```
RP/0/RP0/CPU0:router(config-isis-if-af)# show isis ipv4 route detail
```

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

---

**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
```
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

SUMMARY STEPS

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

**DETAILED STEPS**

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

**Example:**
```
RP/0/RP0/CPU0:router(config)# router isis lsp
```
Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.

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

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

**Example:**
```
RP/0/RP0/CPU0:router(config-isis)# lsp-refresh-interval 10800
```
(Optional) Sets the time between regeneration of LSPs that contain different sequence numbers

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

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

**Example:**
```
RP/0/RP0/CPU0:router(config-isis)# lsp-check-interval 240
```
(Optional) Configures the time between periodic checks of the entire database to validate the checksums of the LSPs in the database.

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

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

**Example:**
```
RP/0/RP0/CPU0:router(config-isis)# lsp-gen-interval maximum-wait 15 initial-wait 5
```
(Optional) Reduces the rate of LSP generation during periods of instability in the network. Helps reduce the CPU load on the router and number of LSP transmissions to its IS-IS neighbors.

- During prolonged periods of network instability, repeated recalculation of LSPs can cause an increased CPU load on the local router. Further, the flooding of these recalculated LSPs to the other Intermediate Systems in the network causes increased traffic and can result in other routers having to spend more time running route calculations.
Step 6  lsp-mtu  bytes  [ level  { 1  |  2  } ]
Example:
RP/0/RP0/CPU0:router(config-isis)# lsp-mtu 1300
(Optional) Sets the maximum transmission unit (MTU) size of LSPs.

Step 7  max-lsp-lifetime  seconds  [ level  { 1  |  2  } ]
Example:
RP/0/RP0/CPU0:router(config-isis)# max-lsp-lifetime 11000
(Optional) Sets the initial lifetime given to an LSP originated by the router.
  • This is the amount of time that the LSP persists in the database of a neighbor unless the LSP is regenerated or refreshed.

Step 8  ignore-lsp-errors  disable
Example:
RP/0/RP0/CPU0:router(config-isis)# ignore-lsp-errors disable
(Optional) Sets the router to purge LSPs received with checksum errors.

Step 9  interface  type interface-path-id
Example:
RP/0/RP0/CPU0:router(config-isis)# interface HundredGigE 0/1/0/3
Enters interface configuration mode.

Step 10  lsp-interval  milliseconds  [ level  { 1  |  2  } ]
Example:
RP/0/RP0/CPU0:router(config-isis-if)# lsp-interval 100
(Optional) Configures the amount of time between each LSP sent on an interface.

Step 11  csnp-interval  seconds  [ level  { 1  |  2  } ]
Example:
RP/0/RP0/CPU0:router(config-isis-if)# csnp-interval 30 level 1
(Optional) Configures the interval at which periodic CSNP packets are sent on broadcast interfaces.
  • Sending more frequent CSNPs means that adjacent routers must work harder to receive them.
  • Sending less frequent CSNP means that differences in the adjacent routers may persist longer.

Step 12  retransmit-interval  seconds  [ level  { 1  |  2  } ]
Example:
RP/0/RP0/CPU0:router(config-isis-if)# retransmit-interval 60
(Optional) Configures the amount of time that the sending router waits for an acknowledgment before it considers that the LSP was not received and subsequently resends.

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

**Step 13**

```
retransmit-throttle-interval milliseconds [ level 1 | 2 ]
```

*Example:*

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

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

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

**Step 14**

```
mesh-group { number | blocked }
```

*Example:*

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

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

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

**Step 15**

```
commit
```

**Step 16**

```
show isis interface [ type interface-path-id | level 1 | 2 ] [ brief ]
```

*Example:*

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

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

**Step 17**

```
show isis [ instance instance-id ] database [ level 1 | 2 ] [ detail | summary | verbose ] [ * | lsp-id ]
```

*Example:*

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

(Optional) Displays the IS-IS LSP database.

**Step 18**

```
show isis [ instance instance-id ] lsp-log [ level 1 | 2 ]
```

*Example:*

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

(Optional) Displays LSP log information.

**Step 19**

```
show isis database-log [ level 1 | 2 ]
```

*Example:*

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

**SUMMARY STEPS**

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

Step 1 configure
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 0/1/0/3

Enters interface configuration mode.

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

Example:

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

Configures the authentication password for an IS-IS interface.
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.

**SUMMARY STEPS**

1. configure
2. router isis instance-id
3. lsp-password keychain keychain-name [ level { 1 | 2 } ] [ send-only ] [ snp send-only ]
4. interface type interface-path-id
5. hello-password keychain keychain-name [ level { 1 | 2 } ] [ send-only ]
6. commit

**DETAILED STEPS**

**Step 1** configure

**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 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 0/1/0/3
```

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  commit

---

**Nonstop Forwarding**

On Cisco IOS XR software, NSF minimizes the amount of time a network is unavailable to its users following a route processor (RP) failover. The main objective of NSF is to continue forwarding IP packets and perform a graceful restart following an RP failover.

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

NSF allows for the forwarding of data packets to continue along known routes while the routing protocol information is being restored following an RP failover. When the NSF feature is configured, peer networking devices do not experience routing flaps. Data traffic is forwarded through intelligent line cards while the standby RP assumes control from the failed active RP during a failover. The ability of line cards to remain up through a failover and to be kept current with the Forwarding Information Base (FIB) on the active RP is key to NSF operation.

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

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

- IETF NSF
- Cisco NSF

If neighbor routers on a network segment are NSF aware, meaning that neighbor routers are running a software version that supports the IETF Internet draft for router restartability, they assist an IETF NSF router that is restarting. With IETF NSF, neighbor routers provide adjacency and link-state information to help rebuild the routing information following a failover.

In Cisco IOS XR software, Cisco NSF checkpoints (stores persistently) all the state necessary to recover from a restart without requiring any special cooperation from neighboring routers. The state is recovered from the neighboring routers, but only using the standard features of the IS-IS routing protocol. This capability makes Cisco NSF suitable for use in networks in which other routers have not used the IETF standard implementation of NSF.
If you configure IETF NSF on the Cisco IOS XR router and a neighbor router does not support IETF NSF, the affected adjacencies flap, but nonstop forwarding is maintained to all neighbors that do support IETF NSF. A restart reverts to a cold start if no neighbors support IETF NSF.

**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.

**SUMMARY STEPS**

1. `configure`
2. `router isis instance-id`
3. `nsf { cisco | ietf }`
4. `nsf interface-expires number`
5. `nsf interface-timer seconds`
6. `nsf lifetime seconds`
7. `commit`
8. `show running-config [ command ]`

**DETAILED STEPS**

**Step 1**

`configure`

**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.
Configure Nonstop Forwarding for IS-IS

- 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

**commit**

### 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 suppresses routing changes on peers to SSO-enabled devices during processor switchover events (SSO or ISSU), reducing network instability and downtime. Non Stop Routing allows for the forwarding of data packets to continue along known routes while the routing protocol information is being restored following a processor switchover. When Non Stop Routing is used, peer networking devices have no knowledge of any event on the switching over router. All information needed to continue the routing protocol peering state is transferred to the standby processor, so it can continue immediately upon a switchover.

Under NSR mode, ISIS processes running on standby sync information with ISIS processes running on active and maintains same set of isis neighbors, lsp databases, tunnels, etc. This reduces the startup time for ISIS process to become fully operational after the switchover, which can be very useful in a scale environment.

Configuring ISIS-NSR

SUMMARY STEPS

1. configure
2. router isis instance-id
3. nsr
4. commit
5. show isis nsr adjacency
6. show isis nsr status
7. show isis nsr statistics

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>router isis instance-id</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config)# router isis 1</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>nsr</td>
<td>Configures the NSR feature.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-isis)# nsr</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>commit</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>show isis nsr adjacency</td>
<td>Displays adjacency information.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router # show isis nsr adjacency</td>
<td>System Id Interface SNPA State Hold Changed NSF</td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4 IPv6</td>
<td></td>
</tr>
<tr>
<td>BFD BFD</td>
<td></td>
</tr>
<tr>
<td>R1-v1S Nii0 <em>PtoP</em> Up 83 00:00:33 Yes None None</td>
<td></td>
</tr>
</tbody>
</table>

### 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
```

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 0 0 0
LIVE INTERFACE: 4 0 4 0
PTP INTERFACE: 1 0 1 0
LAN INTERFACE: 2 0 2 0
LOOPBACK INTERFACE: 1 0 1 0
TE Tunnel: 1 0 1 0
TE LINK: 2 0 2 0
NSR OPTIONAL STATS :
   L1 LSP: 0 0 0 0
   L2 LSP: 4
```

Shows number of ISIS adjacencies, lsp's, routes, tunnels, Te links on active and standby routers.
Configuring IS-IS Adjacency Stagger

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

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

Restrictions

• IS-IS adjacency stagger is only supported on point-to-point interfaces and not on LAN interfaces.

• IS-IS adjacency stagger is not supported with NSF (non-stop forwarding) mechanisms.

Configuration Example

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

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

Configuration

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

/* Configure IS-IS adjacency stagger */
Router(config-isis)# adjacency stagger 2 3
Router(config-isis)# commit
Multiprotocol Label Switching Traffic Engineering

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

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

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

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

Note

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

SUMMARY STEPS

1. configure
2. router isis instance-id
3. address-family { ipv4 | ipv6 } [ unicast ]
4. mpls traffic-eng level { 1 | 2 }
5. mpls traffic-eng router-id { ip-address | interface-name interfacenstance }
6. metric-style wide [ level { 1 | 2 }]
7. commit
8. show isis [ instance instance-id ] mpls traffic-eng tunnel
9. show isis [ instance instance-id ] mpls traffic-eng adjacency-log
10. show isis [ instance instance-id ] mpls traffic-eng advertisements

DETAILED STEPS

Step 1 configure
Step 2 router isis instance-id
Example:

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

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

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

Step 3

address-family { ipv4 | ipv6 } [ unicast ]

Example:

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

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

Step 4

mpls traffic-eng level { 1 | 2 }

Example:

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

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

Step 5

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

Example:

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

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

Step 6

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

Example:

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

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

Step 7

commit

Step 8

show isis [ instance instance-id ] mpls traffic-eng tunnel

Example:

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

(Optional) Displays MPLS TE tunnel information.

Step 9

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

Example:

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

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

Step 10

show isis [ instance instance-id ] mpls traffic-eng advertisements
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.

SUMMARY STEPS

1. configure
2. router isis instance-id
3. log adjacency changes
4. interface type interface-path-id
5. hello-padding { disable | sometimes } [ level { 1 | 2 }]
6. hello-interval seconds [ level { 1 | 2 }]
7. hello-multiplier multiplier [ level { 1 | 2 }]
8. hello-password { hmac-md5 | text } { clear | encrypted } password [ level { 1 | 2 }][ send-only ]
9. commit
10. `show isis [ instance instance-id ] adjacency type interface-path-id [ detail ] [ systemid system-id ]`

11. `show isis adjacency-log`

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

13. `show isis [ instance instance-id ] neighbors [ interface-type interface-instance ] [ summary ] [ detail ] [ systemid system-id ]`

### Detailed Steps

**Step 1**

Configure

**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 0/1/0/3
```

Enters interface configuration mode.

**Step 5**

Hello-paddings ` { 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
```

 specify 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 6
```

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

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  commit

Step 10  show isis [ instance instance-id ] adjacency type 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 0/1/0/1 brief

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

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

Example:

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

(Optional) Displays information about IS-IS neighbors.
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.

SUMMARY STEPS

1. configure
2. router isis instance-id
3. interface type interface-path-id
4. address-family ipv4 unicast
5. mpls ldp sync [ level { 1 | 2 }]
6. commit

DETAILED STEPS

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 0/1/0/3
```

Enters interface configuration mode.

**Step 4**

```
address-family ipv4 unicast
```

**Example:**

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

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

**Step 5**

```
mpls ldp sync [ level { 1 | 2 }]
```

**Example:**

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

Enables MPLS LDP synchronization for the IPv4 address family under interface HundredGigE 0/1/0/3.

**Step 6**

```
commit
```

---

**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

**SUMMARY STEPS**

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

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step 1</th>
<th>configure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2</td>
<td>mpls traffic-eng path-selection ignore overload</td>
</tr>
</tbody>
</table>

**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
```

**Configuring Global Weighted SRLG Protection**

A shared risk link group (SRLG) is a set of links sharing a common resource and thus shares the same risk of failure. The existing loop-free alternate (LFA) implementations in interior gateway protocols (IGPs) support
SRLG protection. However, the existing implementation considers only the directly connected links while computing the backup path. Hence, SRLG protection may fail if a link that is not directly connected but shares the same SRLG is included while computing the backup path. Global weighted SRLG protection feature provides better path selection for the SRLG by associating a weight with the SRLG value and using the weights of the SRLG values while computing the backup path.

To support global weighted SRLG protection, you need information about SRLGs on all links in the area topology. You can flood SRLGs for remote links using ISIS or manually configuring SRLGS on remote links.

**Configuration Examples: Global Weighted SRLG Protection**

There are three types of configurations that are supported for the global weighted SRLG protection feature.

- **local SRLG with global weighted SRLG protection**
- **remote SRLG flooding**
- **remote SRLG static provisioning**

This example shows how to configure the local SRLG with global weighted SRLG protection feature.

```
RP/0/RP0/CPU0:router(config)# srlg
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0
RP/0/RP0/CPU0:router(config-srlg-if)# name group1
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/1
RP/0/RP0/CPU0:router(config-srlg-if)# name group1
RP/0/RP0/CPU0:router(config-srlg)# name group value 100
RP/0/RP0/CPU0:router(config)# router isis 1
RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix srlg-protection weighted-global
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix tiebreaker srlg-disjoint index 1
RP/0/RP0/CPU0:router(config-isis-if-af)# point-to-point
RP/0/RP0/CPU0:router(config-isis-if-af)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
RP/0/RP0/CPU0:router(config-isis-if-srlg)# srlg
RP/0/RP0/CPU0:router(config-isis-srlg)# name group1
RP/0/RP0/CPU0:router(config-isis-srlg-name)# admin-weight 5000
```

This example shows how to configure the global weighted SRLG protection feature with remote SRLG flooding. The configuration includes local and remote router configuration. On the local router, the global weighted SRLG protection is enabled by using the `fast-reroute per-prefix srlg-protection weighted-global` command. In the remote router configuration, you can control the SRLG value flooding by using the `advertise application lfa link-attributes srlg` command. You should also globally configure SRLG on the remote router.

The local router configuration for global weighted SRLG protection with remote SRLG flooding is as follows:

```
RP/0/RP0/CPU0:router(config)# router isis 1
RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix srlg-protection weighted-global
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix tiebreaker srlg-disjoint index 1
RP/0/RP0/CPU0:router(config-isis-if-af)# exit
RP/0/RP0/CPU0:router(config-isis)# interface TenGigE0/0/0
RP/0/RP0/CPU0:router(config-isis-if)# point-to-point
```
The remote router configuration for global weighted SRLG protection with remote SRLG flooding is as follows:

```bash
RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
RP/0/RP0/CPU0:router(config-isis-if-af)# exit
RP/0/RP0/CPU0:router(config-isis)# srlg
RP/0/RP0/CPU0:router(config-isis-srlg)# name group1
RP/0/RP0/CPU0:router(config-isis-srlg-name)# admin-weight 5000

This example shows configuring the global weighted SRLG protection feature with static provisioning of SRLG values for remote links. You should perform these configurations on the local router.

```bash
RP/0/RP0/CPU0:router(config)# srlg
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/0
RP/0/RP0/CPU0:router(config-srlg-if)# name group1
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/1
RP/0/RP0/CPU0:router(config-srlg-if)# name group1
RP/0/RP0/CPU0:router(config-srlg-if)# name group value 100
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config)# router isis 1
RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix srlg-protection weighted-global
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix tiebreaker srlg-disjoint index 1
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
RP/0/RP0/CPU0:router(config-isis)# interface TenGigE0/0/0/0
RP/0/RP0/CPU0:router(config-isis-if)# point-to-point
RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
RP/0/RP0/CPU0:router(config-isis-if-af)# exit
RP/0/RP0/CPU0:router(config-isis)# srlg
RP/0/RP0/CPU0:router(config-isis-srlg)# name group1
RP/0/RP0/CPU0:router(config-isis-srlg-name)# admin-weight 5000
RP/0/RP0/CPU0:router(config-isis-srlg-name)# static ipv4 address 10.0.4.1 next-hop ipv4 address 10.0.4.2
```

References for IS-IS

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

- IS-IS Functional Overview, on page 42
IS-IS Functional Overview

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

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

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

Restrictions

When multiple instances of IS-IS are being run, an interface can be associated with only one instance (process). Instances may not share an interface.

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.

**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 02e9.4522.5526 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.

For more information about the `show isis database` command, see *IS-IS Commands* Chapter of the *Routing Command Reference for Cisco NCS 5500 Series Routers and Cisco NCS 540 Series Routers*.

**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

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

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

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.
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.

1. VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families will be supported in a future release.
2. GTSM TTL Security is not supported.

- Prerequisites for Implementing OSPF, on page 48
- Enable OSPF, on page 48
- Verify OSPF Configuration and Operation, on page 50
- Stub Area, on page 53
- Neighbors and Adjacency for OSPF, on page 56
- Authentication Strategies, on page 60
- Control Frequency That Same LSA Is Originated or Accepted for OSPF, on page 63
- Virtual Link and Transit Area for OSPF, on page 65
- Summarize Subnetwork LSAs on OSPF ABR, on page 70
- Route Redistribution for OSPF, on page 73
- Nonstop Forwarding for OSPF Version 2, on page 75
- OSPF Shortest Path First Throttling, on page 78
- Graceful Restart for OSPFv3, on page 81
- Warm Standby and Nonstop Routing for OSPF Version 2, on page 83
- Warm Standby and Nonstop Routing for OSPF Version 3, on page 84
- OSPFv2OSPF SPF Prefix Prioritization, on page 85
- Configure OSPF as a Provider Edge to Customer Edge (PE-CE) Protocol, on page 90
- Create Multiple OSPF Instances (OSPF Process and a VRF), on page 92
- Label Distribution Protocol IGP Auto-configuration for OSPF, on page 94
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 IP Addresses and Services Configuration Guide for Cisco NCS 5500 Series Routers for information on IPv6 routing and addressing.

• Before you enable OSPFv3 on an interface, you must perform the following tasks:
  • Complete the OSPF network strategy and planning for your IPv6 network. For example, you must decide whether multiple areas are required.
  • Enable IPv6 on the interface.

• Configuring authentication (IP Security) is an optional task. If you choose to configure authentication, you must first decide whether to configure plain text or Message Digest 5 (MD5) authentication, and whether the authentication applies to an entire area or specific interfaces.

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.

SUMMARY STEPS

1. configure
2. Do one of the following:
   • router ospf process-name
   • router ospfv3 process-name
3. router-id { router-id }
4. area area-id
5. interface type interface-path-id
6. Repeat Step 5 for each interface that uses OSPF.
7. log adjacency changes [ detail ] [ enable | disable ]
8. commit
**DETAILED STEPS**

**Step 1**
configure

**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 0/1/0/3
```

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

**Step 6**
Repeat Step 5 for each interface that uses OSPF.
Step 7  

log adjacency changes [ detail ] [ 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  

commit

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.

```plaintext
interface TenGigE 0/3/0/0
    ip address 10.1.2.1 255.255.255.0
    negotiation auto

interface TenGigE 0/3/0/1
    ip address 10.1.3.1 255.255.255.0
    negotiation auto

router ospf 1
    router-id 10.2.3.4
    area 0
        interface TenGigE 0/3/0/0
    area 1
        interface TenGigE 0/3/0/1
```

Verify OSPF Configuration and Operation

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

SUMMARY STEPS

1.  show { ospf | ospfv3 } [ process-name ]
2.  show { ospf | ospfv3 } [ process-name ] border-routers [ router-id ]
3.  show { ospf | ospfv3 } [ process-name ] database
4.  show { ospf | ospfv3 } [ process-name ] [ area-id ] flood-list interface type interface-path-id
5. `show { ospf | ospfv3 } [ process-name ] [ vrf vrf-name ] [ area-id ] interface [ type interface-path-id ]`

6. `show { ospf | ospfv3 } [ process-name ] [ area-id ] neighbor [ type interface-path-id ] [ neighbor-id ] [ detail ]`

7. `clear { ospf | ospfv3 } [ process-name ] process`

8. `clear {ospf|ospfv3} [ process-name ] redistribution`

9. `clear {ospf|ospfv3} [ process-name ] routes`

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

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

**DETAILED STEPS**

**Step 1**

`show { ospf | ospfv3 } [ process-name ]`

*Example:*

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

(Optional) Displays general information about OSPF routing processes.

**Step 2**

`show { ospf | ospfv3 } [ process-name ] border-routers [ router-id ]`

*Example:*

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

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

**Step 3**

`show { ospf | ospfv3 } [ process-name ] database`

*Example:*

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

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

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

**Step 4**

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

*Example:*

RP/0/RP0/CPU0:router# show ospf 100 flood-list interface TenGigE 0/3/0/0

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

**Step 5**

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

*Example:*

RP/0/RP0/CPU0:router# show ospf 100 interface TenGigE 0/3/0/0

*Example:*
Verify OSPF Configuration and Operation

Implementing OSPF

Verify OSPF Configuration and Operation

Step 6

```
show { ospf | ospfv3 } [[ process-name ] [ area-id ] neighbor [ type 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/RP0/CPU0:router# clear ospf 100 process
```

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

Step 8

```
clear { ospf | ospfv3 } [ process-name ] redistribution
```

**Example:**

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

Cleans OSPF route redistribution.

Step 9

```
clear { ospf | ospfv3 } [ process-name ] routes
```

**Example:**

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

Cleans 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 ]
```

Routing Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
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. The Type 7 LSAs are translated into Type 5 LSAs by NSSA ABRs, which are flooded throughout the whole routing domain. Summarization and filtering are supported during the translation.

Use NSSA to simplify administration if you are a network administrator that must connect a central site using OSPF to a remote site that is using a different routing protocol.

Before NSSA, the connection between the corporate site border router and remote router could not be run as an OSPF stub area because routes for the remote site could not be redistributed into a stub area, and two routing protocols needed to be maintained. A simple protocol like RIP was usually run and handled the redistribution. With NSSA, you can extend OSPF to cover the remote connection by defining the area between the corporate router and remote router as an NSSA. Area 0 cannot be an NSSA.

Configure Stub and Not-So-Stubby Area Types

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

SUMMARY STEPS

1. configure
2. Do one of the following:
   - router ospf process-name
   - router ospfv3 process-name
3. `router-id { router-id }`
4. `area area-id`
5. Do one of the following:
   - `stub [ no-summary ]`
   - `nssa [ no-redistribution ] [ default-information-originate ] [ no-summary ]`
6. Do one of the following:
   - `stub`
   - `nssa`
7. `default-cost cost`
8. `commit`
9. Repeat this task on all other routers in the stub area or NSSA.

**DETAILED STEPS**

**Step 1**
configure

**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 1
Enters area configuration mode and configures a nonbackbone area for the OSPF process.

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

**Step 5**
Do one of the following:

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

**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 6**
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 7**
**default-cost**  **cost**

**Example:**

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

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

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

Step 8 commit
Step 9 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:

```plaintext
router ospfv3 1
  router-id 10.0.0.217
  area 0
    interface TenGigE 0/2/0/1
    area 1
      stub
      interface TenGigE 0/2/0/0
```

**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.

**SUMMARY STEPS**

1. configure
2. Do one of the following:
   - `router ospf` `process-name`
   - `router ospfv3` `process-name`
3. `router-id` `{ router-id }
4. `area` `area-id`
5. `network` `{ broadcast | non-broadcast }`
6. **dead-interval** seconds
7. **hello-interval** seconds
8. **interface** type interface-path-id
9. Do one of the following:
   - **neighbor** ip-address [priority number] [poll-interval seconds] [cost number]
   - **neighbor** ipv6-link-local-address [priority number] [poll-interval seconds] [cost number]
   - [database-filter] [all]
10. Repeat Step 9 for all neighbors on the interface.
11. **exit**
12. **interface** type interface-path-id
13. Do one of the following:
   - **neighbor** ip-address [priority number] [poll-interval seconds] [cost number]
     [database-filter] [all]
   - **neighbor** ipv6-link-local-address [priority number] [poll-interval seconds] [cost number]
     [database-filter] [all]
14. Repeat Step 13 for all neighbors on the interface.
15. **commit**

**DETAILED STEPS**

**Step 1**
configure

**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 0/2/0/0

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

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

Step 9  Do one of the following:

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

Example:
Configure Neighbors for Nonbroadcast Networks

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

```
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
```

Configure 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 0/3/0/0
```
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:f3fe
```
Configure the IPv4 address of OSPF neighbors interconnecting to nonbroadcast networks.

or

Configure 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 “black-holing” or routing loops.

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

**Step 15**  
`commit`

---

**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 105 for information about each type of authentication and when you should use a specific method for your network.
SUMMARY STEPS

1. configure
2. router ospf  process-name
3. router-id { router-id }
4. authentication [ message-digest | null ]
5. message-digest-key  key-id  md5 { key | clear key | encrypted key | LINE}
6. area  area-id
7. interface  type interface-path-id
8. Repeat Step 7 for each interface that must communicate, using the same authentication.
9. exit
10. area  area-id
11. authentication [ message-digest | null ]
12. interface  type interface-path-id
13. Repeat Step 12 for each interface that must communicate, using the same authentication.
14. interface  type interface-path-id
15. authentication [ message-digest | null ]
16. commit

DETAILED STEPS

Step 1  configure
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 0/1/0/3

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

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

Step 8

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

Step 9

exit

Example:

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

Enters area OSPF configuration mode.

Step 10

area area-id

Example:

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

Enters area configuration mode and configures a nonbackbone area 1 for the OSPF process.

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

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 0/1/0/0  
Enter 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 0/3/0/0  
Enter 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 0/3/0/0, overriding the plain text authentication specified for area 1.  
• By default, all of the interfaces configured in the same area inherit the same authentication parameter values of the area.

Step 16  
commit

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.

SUMMARY STEPS

1. configure  
2. Do one of the following:  
   • router ospf  process-name  
   • router ospfv3  process-name  
3. router-id  { router-id }
4. Perform Step 5 or Step 6 or both to control the frequency that the same LSA is originated or accepted.
5. `timers lsa refresh seconds`
6. `timers lsa min-arrival seconds`
7. `timers lsa group-pacing seconds`
8. `commit`

**DETAILED STEPS**

**Step 1** configure

**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  commit

---

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 1: 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.

SUMMARY STEPS

1. Do one of the following:

  • show ospf [ process-name ]
• show ospfv3 [process-name]

2. configure
3. Do one of the following:
   • router ospf process-name
   • router ospfv3 process-name
4. router-id {router-id}
5. area area-id
6. virtual-link router-id
7. authentication message-digest
8. message-digest-key key-id md5 {key | clear key | encrypted key}
9. Repeat all of the steps in this task on the ABR that is at the other end of the virtual link. Specify the same key ID and key that you specified for the virtual link on this router.
10. commit
11. Do one of the following:
   • show ospf [process-name][area-id] virtual-links
   • show ospfv3 [process-name] virtual-links

DETAILED STEPS

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

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:**

```
RP/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**
`commit`

**Step 11**
Do one of the following:

• `show ospf [process-name] [area-id] virtual-links`

• `show ospfv3 [process-name] virtual-links`

**Example:**

```bash
RP/0/CPU0:router# show ospf 1 2 virtual-links
```

or

```bash
RP/0/CPU0:router# show ospfv3 1 virtual-links
```

(Optional) Displays the parameters and the current state of OSPF virtual links.

---

**Creating virtual link - example**

**ABR 1 Configuration**

**ABR 2 Configuration**

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.

```bash
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 0/1/0/1 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 0/2/0/1
    area 1
    virtual-link 10.0.0.212
    interface TenGigE 0/2/0/0

router ospfv3 1
  router-id 10.0.0.212
  area 0
    interface TenGigE 0/3/0/1
    area 1
    virtual-link 10.0.0.217
    interface TenGigE 0/2/0/0
```

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 0/2/0/1
    interface TenGigE 0/3/0/0
  area 1
    interface TenGigE 0/2/0/0
    virtual-link 10.10.5.5

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 0/9/0/1
    virtual-link 10.10.10.10
  area 3
    interface Loopback 0
    interface TenGigE 0/9/0/0
```

### 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.

**SUMMARY STEPS**

1. configure
2. Do one of the following:
   - `router ospf  process-name`
   - `router ospfv3  process-name`
3. `router-id  { router-id }
4. area  area-id`
5. Do one of the following:
   - `range  ip-address mask  [ advertise | not-advertise ]`
   - `range  ipv6-prefix /prefix-length  [ advertise | not-advertise ]`
6. `interface  type interface-path-id`
7. commit

**DETAILED STEPS**

**Step 1** configure

**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:**

```bash
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:**

```bash
RP/0/RP0/CPU0:router(config-ospf-ar)# range 192.168.0.0 255.255.0.0 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:**

```bash
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**

`commit`

**Example**

The following example shows the prefix range 2300::/16 summarized from area 1 into the backbone:
**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.

**SUMMARY STEPS**

1. configure
2. Do one of the following:
   - `router ospf` `process-name`
   - `router ospfv3` `process-name`
3. `router-id` `{ router-id }`
4. `redistribute` `protocol` `{ process-id }` `{ level-1 | level-1-2 | level-2 }` `{ metric metric-value }` `{ metric-type type-value }` `{ match { external { 1 | 2 } } | tag tag-value }` `{ route-policy policy-name }`
5. Do one of the following:
   - `summary-prefix address mask` `{ not-advertise }` `{ tag tag }`
   - `summary-prefix ipv6-prefix / prefix-length` `{ not-advertise }` `{ tag tag }`
6. commit

**DETAILED STEPS**

**Step 1** configure

**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
```
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  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  
commit

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 0/2/0/0

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.

---

**SUMMARY STEPS**

1. configure
2. router ospf  process-name
3. router-id  { router-id }
4. Do one of the following:
   - nsf cisco
   - nsf cisco  enforce global
5. `nssf interval  seconds`
6. `nsfflush-delay-time seconds`
7. `nsflifetime seconds`
8. `nsfietf`
9. `commit`

**DETAILED STEPS**

**Step 1**
configure

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

*Example:*
```
RP/0/RP0/CPU0:router(config)# router ospf 1
```
Enables OSPF routing for the specified routing process and places the router in router configuration mode.

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

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

*Example:*
```
RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3
```
Configures a router ID for the OSPF process.

*Note*  We recommend using a stable IPv4 address as the router ID.

**Step 4**
Do one of the following:

- `nsf cisco`
- `nsf cisco  enforce global`

*Example:*
```
RP/0/RP0/CPU0:router(config-ospf)# nsf cisco enforce global
```
Enables Cisco NSF operations for the OSPF process.

- Use the `nsf cisco` command without the optional `enforce` and `global` keywords to abort the NSF restart mechanism on the interfaces of detected non-NSF neighbors and allow NSF neighbors to function properly.
- Use the `nsf cisco` command with the optional `enforce` and `global` keywords if the router is expected to perform NSF during restart. However, if non-NSF neighbors are detected, NSF restart is canceled for the entire OSPF process.

**Step 5**
`nsf interval  seconds`

*Example:*
```
RP/0/RP0/CPU0:router(config-ospf)# nsf interval 120
```
Sets the minimum time between NSF restart attempts.
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**  
*commit*

---

**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 2: 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 3: Timer Intervals Reset After Topology Change Event, on page 79 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 3: 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.

**SUMMARY STEPS**

1. `configure`
2. Do one of the following:
   - `router ospf  process-name`
   - `router ospfv3  process-name`
3. `router-id  { router-id }`
4. `timers throttle spf  spf-start spf-hold spf-max-wait`
5. `area  area-id`
6. `interface  type interface-path-id`
7. `commit`
8. Do one of the following:
   - `show ospf  [ process-name ]`
   - `show ospfv3  [ process-name ]`
DETAILED STEPS

Step 1  configure

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  commit

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# 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:

• RP failure resulting in a switch-over to the backup processor
• Planned OSPFv3 process restart, such as a restart resulting from a software upgrade or downgrade
• Unplanned OSPFv3 process restart, such as a restart resulting from a process crash

In addition, OSPFv3 will unilaterally shutdown and enter the exited state when a critical memory event, indicating the processor is critically low on available memory, is received from the sysmon watch dog process.

This feature supports nonstop data forwarding on established routes while the OSPFv3 routing protocol restarts. Therefore, this feature enhances high availability of IPv6 forwarding.

Configure OSPFv3 Graceful Restart

This task explains how to configure a graceful restart for an OSPFv3 process. This task is optional.

SUMMARY STEPS

1. configure
2. router ospfv3  process-name
3. graceful-restart
4. graceful-restart lifetime
5. graceful-restart interval  seconds
6. graceful-restart helper disable
7. commit
8. show ospfv3  [ process-name [ area-id ] ] database grace
DETAILED STEPS

Step 1 configure
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 commit
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 SPF s 10000 msecs
Maximum wait time between two consecutive SPF s 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
```

Warm Standby and Nonstop Routing for OSPF Version 2

OSPFv2 warm standby provides high availability across RP switchovers. With warm standby extensions, each process running on the active RP has a corresponding standby process started on the standby RP. A standby OSPF process can send and receive OSPF packets with no performance impact to the active OSPF process.

Nonstop routing (NSR) allows an RP failover, process restart, or in-service upgrade to be invisible to peer routers and ensures that there is minimal performance or processing impact. Routing protocol interactions between routers are not impacted by NSR. NSR is built on the warm standby extensions. NSR alleviates the requirement for Cisco NSF and IETF graceful restart protocol extensions.
NSR is enabled by default for OSPF. To disable NSR, use the `nsr disable` command in the OSPF configuration mode.

Note: It is recommended to set the Hello interval value to a default of 10 seconds. OSPF sessions may flap during switchover if the Hello interval value configured is less than the default value.

**Enable Nonstop Routing for OSPFv2**

This optional task describes how to enable nonstop routing (NSR) for OSPFv2 process. NSR is disabled by default. When NSR is enabled, OSPF process on the active RP synchronizes all necessary data and states with the OSPF process on the standby RP. When the switchover happens, OSPF process on the newly active RP has all the necessary data and states to continue running and does not require any help from its neighbors.

**SUMMARY STEPS**

1. configure
2. router ospf instance-id
3. nsr
4. commit

**DETAILED STEPS**

**Step 1** configure
**Step 2** router ospf instance-id

**Example:**
```
RP/0/RP0/CPU0:router(config)# router ospf isp
```

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

**Step 3** nsr

**Example:**
```
RP/0/RP0/CPU0:router(config-ospf)# nsr
```

Enables NSR for the OSPFv2 process.

**Step 4** commit

**Warm Standby and Nonstop Routing for OSPF Version 3**

This feature helps OSPFv3 to initialize itself prior to Fail over (FO) and be ready to function before the failure occurs. It reduces the downtime during switchover. By default, the router sends hello packets every 40 seconds.
With warm standby process for each OSPF process running on the Active Route Processor, the corresponding OSPF process must start on the Standby RP. There are no changes in configuration for this feature.

Warm-Standby is always enabled. This is an advantage for the systems running OSPFv3 as their IGP when they do RP failover.

Enable Nonstop Routing for OSPFv3

This task describes how to enable nonstop routing (NSR) for OSPFv3 process. NSR is disabled by default. When NSR is enabled, OSPF process on the active RP synchronizes all necessary data and states with the OSPF process on the standby RP. When the switchover happens, OSPF process on the newly active RP has all the necessary data and states to continue running and does not require any help from its neighbors.

SUMMARY STEPS

1. configure
2. router ospfv3 instance-id
3. nsr
4. commit

DETAILLED STEPS

Step 1 configure
Step 2 router ospfv3 instance-id

Example:

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

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

Step 3 nsr

Example:

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

Enables NSR for the OSPFv3 process.

Step 4 commit

OSPFv2 OSPF 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.

**SUMMARY STEPS**

1. configure
2. prefix-set prefix-set name
3. route-policy route-policy name if destination in prefix-set name then set spf-priority {critical | high | medium} endif
4. Use one of these commands:
   - router ospf ospf-name
   - router ospfv3 ospfv3-name
5. router ospf ospf name
6. spf prefix-priority route-policy route-policy name
7. commit
8. show rpl route-policy route-policy name detail

**DETAILED STEPS**

**Step 1** configure

**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

```
commit
```

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
endif
end-policy

router ospf 1
  spf prefix-priority route-policy ospf-priority
  area 0
    interface TenGigE 0/3/0/0
      !
    area 3
      interface TenGigE 0/2/0/0
        !
    area 8
      interface TenGigE 0/2/0/0

router ospfv3 1
  spf prefix-priority route-policy ospf-priority
  area 0
    interface TenGigE 0/3/0/0
      !
    area 3
      interface TenGigE 0/2/0/0
        !
```
Configure OSPF as a Provider Edge to Customer Edge (PE-CE) Protocol

SUMMARY STEPS

1. configure
2. router ospf process-name
3. vrf vrf-name
4. router-id { router-id }
5. redistribute protocol { process-id } { level-1 | level-1-2 | level-2 } [ metric metric-value ] [ metric-type type-value ] [ match { external [ 1 | 2 ] } ] [ tag tag-value ] route-policy policy-name
6. area area-id
7. interface type interface-path-id
8. exit
9. domain-id [ secondary ] type { 0005 | 0105 | 0205 | 8005 } value value
10. domain-tag tag
11. disable-dn-bit-check
12. commit

DETAILED STEPS

Step 1 configure
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:
Configure OSPF as a Provider Edge to Customer Edge (PE-CE) Protocol

Configure 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] { 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-vrf)# redistribute bgp 1 level-1
```

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 0/2/0/0
```

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
```
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.

SUMMARY STEPS

1. configure
2. router ospf  process-name
3. area  area-id
4. interface type interface-path-id
5. exit
6. vrf  vrf-name
7. area  area-id
8. interface type interface-path-id
9. commit

DETAILED STEPS

Step 1  configure

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:**

```plaintext
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:**

```plaintext
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/1/0/3
```

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

**Step 5**  exit

**Example:**

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

Enters OSPF configuration mode.

**Step 6**  vrf  *vrf-name*

**Example:**

```plaintext
RP/0/RP0/CPU0:router(config-ospf)# vrf vrf1
```

Creates a VRF instance and enters VRF configuration mode.

**Step 7**  area  *area-id*

**Example:**

```plaintext
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:**

```plaintext
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.
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.

**SUMMARY STEPS**

1. configure
2. router ospf  process-name
3. mpls ldp auto-config
4. commit

**DETAILED STEPS**

**Step 1** configure

**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.
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.

### SUMMARY STEPS

1. **configure**
2. **router ospf** process-name
3. (Optional) **vrf** vrf-name
4. Use one of the following commands:
   - mpls ldp sync
   - area area-id mpls ldp sync
   - area area-id interface name mpls ldp sync
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
6. **commit**
7. (Optional) **show mpls ldp vrf** vrf-name ipv4 igp sync
8. (Optional) **show mpls ldp vrf** all ipv4 igp sync
9. (Optional) **show mpls ldp** { ipv4 | ipv6 } igp sync

### DETAILED STEPS

**Step 1**
**configure**

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

**Example:**

```
RP/0/RP0/CPU0:router(config)# router ospf 100
```

Identifies the OSPF routing process and enters OSPF configuration mode.

**Step 3**
(Optional) **vrf** vrf-name

**Example:**
Configure LDP IGP Synchronization: OSPF

```
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**

`commit`

**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.

**SUMMARY STEPS**

1. configure
2. router ospf  process-name
3. router-id  { router-id }
4. area  area-id
5. interface  type interface-path-id
6. authentication  [ message-digest keychain  |  keychain keychain | null ]
7. commit
DETAILED STEPS

Step 1  configure
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
```

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
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
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_int1 keychain must be configured before you attempt this step.

Step 7 commit

Examples
The following example shows how to configure the keychain ospf_intf_1 that contains five key IDs. Each key ID is configured with different send-lifetime values; however, all key IDs specify the same text string for the key.

key chain ospf_intf_1
key 1
send-lifetime 11:30:30 May 1 2007 duration 600
cryptographic-algorithm MD5T
key-string clear ospf_intf_1
key 2
send-lifetime 11:40:30 May 1 2007 duration 600
cryptographic-algorithm MD5
key-string clear ospf_intf_1
key 3
send-lifetime 11:50:30 May 1 2007 duration 600
cryptographic-algorithm MD5
key-string clear ospf_intf_1
key 4
send-lifetime 12:00:30 May 1 2007 duration 600
cryptographic-algorithm MD5
key-string clear ospf_intf_1
key 5
send-lifetime 12:10:30 May 1 2007 duration 600
cryptographic-algorithm MD5
key-string clear ospf_intf_1

The following example shows that keychain authentication is enabled on the TenGigE 0/0/0/0 interface:

show ospf 1 interface TenGigE 0/0/0/0
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.

**SUMMARY STEPS**

1. configure
2. router ospf  process-name
3. router-id  { router-id }
4. log adjacency changes  [ detail | disable ]
5. nsf  { cisco  [ enforce global ] | ietf  [ helper disable ] }
6. timers throttle spf  spf-start spf-hold spf-max-wait
7. area  area-id
8. interface  type  interface-path-id
9. security ttl  [ disable | hops  hop-count ]
10. commit
11. show ospf  [ process-name ]  [ area-id ]  interface  [ type  interface-path-id ]

**DETAILED STEPS**

**Step 1**

configure

**Example:**

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

**Step 2**

router ospf  process-name

**Example:**

```bash
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:**

```bash
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:**

```bash
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 TenGigE0/5/0/0
```

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

**Step 9**  
`security ttl [ disable | hops hop-count ]`

Example:

```
RP/0/RP0/CPU0:router(config-ospf-ar-if)# security ttl hopes 2
```

Sets the security TTL value in the IP header for OSPF packets.

**Step 10**  
`commit`

**Step 11**  
`show ospf [ process-name ] [ area-id ] interface [ type interface-path-id ]`

Example:

```
RP/0/RP0/CPU0:router# show ospf 1 interface TenGigE0/5/0/0
```

-routing configuration guide for cisco ncs 5500 series routers, ios xr release 6.3.x
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

TenGigE0/5/0/0 is up, line protocol is up
Internet Address 120.10.10.1/24, Area 0
Process ID 1, Router ID 100.100.100.100, Network Type BROADCAST, Cost: 1
Transmit Delay is 1 sec, State BDR, Priority 1
TTL security enabled, hop count 2
Designated Router (ID) 102.102.102.102, Interface address 120.10.10.1
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
```

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

Routing Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
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.

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 4: 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 4: OSPF Routing Components, on page 105 shows two autonomous systems: 109 and 65200. An autonomous system can consist of one or more OSPF areas.

Areas

Areas allow the subdivision of an autonomous system into smaller, more manageable networks or sets of adjacent networks. As shown in Figure 4: OSPF Routing Components, on page 105, 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 4: OSPF Routing Components, on page 105, 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 4: OSPF Routing Components, on page 105, 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 4: OSPF Routing Components, on page 105) 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).

OSPF Version 3 supports all types of authentication except key rollover.

**Plain Text Authentication**

Plain text authentication (also known as Type 1 authentication) uses a password that travels on the physical medium and is easily visible to someone that does not have access permission and could use the password to infiltrate a network. Therefore, plain text authentication does not provide security. It might protect against a faulty implementation of OSPF or a misconfigured OSPF interface trying to send erroneous OSPF packets.

**MD5 Authentication**

MD5 authentication provides a means of security. No password travels on the physical medium. Instead, the router uses MD5 to produce a message digest of the OSPF packet plus the key, which is sent on the physical medium. Using MD5 authentication prevents a router from accepting unauthorized or deliberately malicious routing updates, which could compromise your network security by diverting your traffic.

---

**Note**

MD5 authentication supports multiple keys, requiring that a key number be associated with a key.
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 mode.

**Restart Mode**

When the OSPFv3 process starts up, it determines whether it must attempt a graceful restart. The determination is based on whether graceful restart was previously enabled. (OSPFv3 does not attempt a graceful restart upon the first-time startup of the router.) When OSPFv3 graceful restart is enabled, it changes the purge timer in the RIB to a nonzero value.

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
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.

Note
VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families will be supported in a future release.

- Verify RIB Configuration Using Routing Table, on page 117
- Verify Networking and Routing Problems, on page 118
- Disable RIB Next-hop Dampening, on page 120
- Enable RCC and LCC On-demand Scan, on page 121
- Enable RCC and LCC Background Scan, on page 122
- References for RIB, on page 124

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.

SUMMARY STEPS

1. show route [vrf {vrf-name | all}] [afi-all | ipv4 | ipv6] [unicast | safi-all] summary [detail] [standby]
2. show route [vrf {vrf-name | all}] [afi-all | ipv4 | ipv6] [unicast | safi-all] [protocol [instance] | ip-address mask] [standby] [detail]
DETAILED STEPS

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
       l - IS-IS inter area, u - IS-IS summary null, * - candidate default
       G - OSPF external type 1, C - CONEXT
   172.73.51.0/24 is directly connected, 2d20h, HundredGigE 4/0/0/1
     Backup O E2 [110/1] via 10.12.12.2, HundredGigE 3/0/0/1
```

Verify Networking and Routing Problems

Perform this task to verify the operation of routes between nodes.

SUMMARY STEPS

1. show route [vrf {vrf-name | all}] [afi-all | ipv4 | ipv6] [unicast | safi-all] [protocol [instance] | ip-address mask] [standby] [detail]
2. show route [ vrf { vrf-name | all }][ afi-all | ipv4 | ipv6 ][ unicast | safi-all ] backup [ ip-address ] [ standby ]

3. show route [ vrf { vrf-name | all }][ ipv4 | ipv6 ][ unicast | safi-all ] best-local [ ip-address ] [ standby ]

4. show route [ vrf { vrf-name | all }][ afi-all | ipv4 | ipv6 ][ unicast | safi-all ] connected [ standby ]

5. show route [ vrf { vrf-name | all }][ afi-all | ipv4 | ipv6 ][ unicast | safi-all ] local [ interface ] [ standby ]

6. show route [ vrf { vrf-name | all }][ ipv4 | ipv6 ][ unicast | safi-all ] longer-prefixes { ip-address mask | ip-address / prefix-length } [ standby ]

7. show route [ vrf { vrf-name | all }][ ipv4 | ipv6 ][ unicast | safi-all ] next-hop [ ip-address ] [ standby ]

**DETAILED STEPS**

**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 192.168.1.11/8
```

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**

```bash
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**

```bash
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, Ethernet0/1/0/0
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.

**SUMMARY STEPS**

1. router rib
2. address-family { ipv4 | ipv6 } next-hop dampening disable
3. commit
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.

SUMMARY STEPS

1. Use one of these commands.
   • show rcc {ipv4 | ipv6} unicast [all] [prefix/mask] [vrf vrf-name]
   • show lcc {ipv4 | ipv6} unicast [all] [prefix/mask] [vrf vrf-name]

2. Use one of these commands.
Enable RCC and LCC Background Scan

Perform this task to run a background scan for Route Consistency Checker (RCC) and Label Consistency Checker (LCC).

**SUMMARY STEPS**

1. configure
2. Use one of these commands:
   - `rcc {ipv4 | ipv6} unicast {enable | period milliseconds}`
   - `lcc {ipv4 | ipv6} unicast {enable | period milliseconds}`
3. commit
4. Use one of these commands.
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:
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 vrf 1:

```
show rcc ipv4 unicast 10.10.0.0/16 vrf vrf1
```

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 124
- RIB Administrative Distance, on page 124
- RIB Statistics, on page 125
- RIB Quarantining, on page 126
- Route and Label Consistency Checker, on page 126

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 3: Default Administrative Distances**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Administrative Distance Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected or local routes</td>
<td>0</td>
</tr>
<tr>
<td>Static routes</td>
<td>1</td>
</tr>
<tr>
<td>External BGP routes</td>
<td>20</td>
</tr>
<tr>
<td>OSPF routes</td>
<td>110</td>
</tr>
<tr>
<td>IS-IS routes</td>
<td>115</td>
</tr>
<tr>
<td>Internal BGP routes</td>
<td>200</td>
</tr>
</tbody>
</table>

The administrative distance for some routing protocols (for instance IS-IS, OSPF, and BGP) can be changed. See the protocol-specific documentation for the proper method to change the administrative distance of that protocol.

---

**Note**

Changing the administrative distance of a protocol on some but not all routers can lead to routing loops and other undesirable behavior. Doing so is not recommended.

---

**RIB 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.
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

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:

---

**Note**

- Currently, only default VRF is supported. L3VPN, VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families and Multicast will be supported in a future release.

- Restrictions for Implementing Routing Policy, on page 129
- Define Route Policy, on page 130
- Attach Routing Policy to BGP Neighbor, on page 131
- Modify Routing Policy Using Text Editor, on page 132
- References for Routing Policy, on page 135
• 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.

## 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.

### SUMMARY STEPS

1. configure
2. route-policy name [ parameter1, parameter2, …, parameterN ]
3. end-policy
4. commit

### DETAILED STEPS

#### Step 1
configure

#### 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  commit

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.

```bash
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.

SUMMARY STEPS

1. configure
2. router bgp  as-number
3. neighbor  ip-address
Modify Routing Policy Using Text Editor

This task explains how to modify an existing routing policy using a text editor.

DETAILED STEPS

Step 1 configure
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 | ipv4 labeled-unicast | ipv4 tunnel | ipv4 mdt | ipv6 unicast | ipv6 labeled-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 commit
SUMMARY STEPS

1. edit { route-policy | prefix-set | as-path-set | community-set | extcommunity-set { rt | soo } | policy-global | rd-set } name [ nano | emacs | vim | inline | add | prepend | remove } set-element

2. show rpl route-policy [ name [ detail ] | states | brief ]

3. show rpl prefix-set [ name | states | brief ]

DETAILED STEPS

---

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.

```plaintext
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

routing 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
```

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.

```plaintext
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:

```plaintext
  as-path-set aset1
  ios-regex '_42$',
  ios-regex '_127$'
end-set
```

This AS path set comprises two elements. When used in a matching operation, this AS path set matches any route whose AS path ends with either the autonomous system (AS) number 42 or 127.

To remove the named AS path set, use the `no as-path-set aset1` command-line interface (CLI) command.

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:

```plaintext
  (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)
($ipaddr:666, 1234:$tag, 1.2.3.4:777, $tag2:777)

These options are supported under extended community set RT:

RP/0/RP0/CPU0:router(config)#extcommunity-set rt rt_set
RP/0/RP0/CPU0:router(config-ext)#?

#-remark  Remark beginning with '#'
*          Wildcard (any community or part thereof)
<1-4294967295> 32-bit decimal number
<1-65535>  16-bit decimal number
A.B.C.D/M:N  Extended community - IPv4 prefix format
A.B.C.D:N   Extended community - IPv4 format
ASN:N       Extended community - ASPLAIN format
X.Y:N       Extended community - ASDOT format
abort       Discard RPL definition and return to top level config
dfa-regex   DFA style regular expression
end-set     End of set definition
exit        Exit from this submode
ios-regex   Traditional IOS style regular expression
show        Show partial RPL configuration

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#-remark</td>
<td>Remark beginning with '##'</td>
</tr>
<tr>
<td>*</td>
<td>Wildcard (any community or part thereof)</td>
</tr>
<tr>
<td>&lt;1-4294967295&gt;</td>
<td>32-bit decimal number</td>
</tr>
<tr>
<td>&lt;1-65535&gt;</td>
<td>16-bit decimal number</td>
</tr>
<tr>
<td>A.B.C.D/M:N</td>
<td>Extended community - IPv4 prefix format</td>
</tr>
<tr>
<td>A.B.C.D:N</td>
<td>Extended community - IPv4 format</td>
</tr>
<tr>
<td>ASN:N</td>
<td>Extended community - ASPLAIN format</td>
</tr>
<tr>
<td>X.Y:N</td>
<td>Extended community - ASDOT format</td>
</tr>
<tr>
<td>abort</td>
<td>Discard RPL definition and return to top level config</td>
</tr>
<tr>
<td>dfa-regex</td>
<td>DFA style regular expression</td>
</tr>
<tr>
<td>end-set</td>
<td>End of set definition</td>
</tr>
<tr>
<td>exit</td>
<td>Exit from this submode</td>
</tr>
<tr>
<td>ios-regex</td>
<td>Traditional IOS style regular expression</td>
</tr>
<tr>
<td>show</td>
<td>Show partial RPL configuration</td>
</tr>
</tbody>
</table>
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-ext)# 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
dnf-set        End of set definition
exit           Exit from this submode
ios-regex      Traditional IOS style regular expression
show           Show partial RPL configuration
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#-remark</td>
<td>Remark beginning with '#'</td>
</tr>
<tr>
<td>*</td>
<td>Wildcard (any community or part thereof)</td>
</tr>
<tr>
<td>&lt;1-4294967295&gt;</td>
<td>32-bit decimal number</td>
</tr>
<tr>
<td>&lt;1-65535&gt;</td>
<td>16-bit decimal number</td>
</tr>
<tr>
<td>A.B.C.D/M:N</td>
<td>Extended community - IPv4 prefix format</td>
</tr>
<tr>
<td>A.B.C.D:N</td>
<td>Extended community - IPv4 format</td>
</tr>
<tr>
<td>ASN:N</td>
<td>Extended community - ASPLAIN format</td>
</tr>
<tr>
<td>X.Y:N</td>
<td>Extended community - ASDOT format</td>
</tr>
<tr>
<td>abort</td>
<td>Discard RPL definition and return to top level config</td>
</tr>
<tr>
<td>dfa-regex</td>
<td>DFA style regular expression</td>
</tr>
<tr>
<td>end-set</td>
<td>End of set definition</td>
</tr>
<tr>
<td>exit</td>
<td>Exit from this submode</td>
</tr>
<tr>
<td>ios-regex</td>
<td>Traditional IOS style regular expression</td>
</tr>
<tr>
<td>show</td>
<td>Show partial RPL configuration</td>
</tr>
</tbody>
</table>
prefix-set

A prefix-set holds IPv4 or IPv6 prefix match specifications, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. The address is a standard dotted-decimal IPv4 or colon-separated hexadecimal IPv6 address. The mask length, if present, is a nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6) following the address and separated from it by a slash. The optional minimum matching length follows the address and optional mask length and is expressed as the keyword ge (mnemonic for greater than or equal to), followed by a nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6). The optional maximum matching length follows the rest and is expressed by the keyword le (mnemonic for less than or equal to), followed by yet another nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6). A syntactic shortcut for specifying an exact length for prefixes to match is the eq keyword (mnemonic for equal to).

If a prefix match specification has no mask length, then the default mask length is 32 for IPv4 and 128 for IPv6. The default minimum matching length is the mask length. If a minimum matching length is specified, then the default maximum matching length is 32 for IPv4 and 128 for IPv6. Otherwise, if neither minimum nor maximum is specified, the default maximum is the mask length.

The prefix-set itself is a comma-separated list of prefix match specifications. The following are examples:

```
prefix-set legal-ipv4-prefix-examples
  10.0.1.1,
  10.0.2.0/24,
  10.0.3.0/24 ge 28,
  10.0.4.0/24 le 28,
  10.0.5.0/24 ge 26 le 30,
  10.0.6.0/24 eq 28,
  10.0.7.2/32 ge 16 le 24,
  10.0.8.0/26 ge 8 le 16
end-set
```

```
prefix-set legal-ipv6-prefix-examples
  2001:0:0:1::/64,
  2001:0:0:2::/64 ge 96,
  2001:0:0:2::/64 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 10.0.3.0/24 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
```
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 hold 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:

```plaintext
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 Policy

The following example shows how the policy accepts all presented routes without modifying the routes.

```plaintext
route-policy quickstart-pass
  pass
end-policy
```

Drop Everything Policy

The following example shows how the policy explicitly rejects all routes presented to it. This type of policy is used to ignore everything coming from a specific peer.

```plaintext
route-policy quickstart-drop
  drop
end-policy
```

Ignore Routes with Specific AS Numbers in the Path

The following example shows the policy definition in three parts. First, the `as-path-set` command defines three regular expressions to match against an AS path. Second, the `route-policy` command applies the AS path set to a route. If the AS path attribute of the route matches the regular expression defined with the `as-path-set` command, the protocol refuses the route. Third, the route policy is attached to BGP neighbor 10.0.1.2. BGP consults the policy named `ignore_path_as` on routes received (imported) from neighbor 10.0.1.2.

```plaintext
as-path-set ignore_path
  ios-regex '_11_',
  ios-regex '_22_',
  ios-regex '_33_'
end-set

route-policy ignore_path_as
  if as-path in ignore_path then
    drop
```

Routing Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
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
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 glbpathtype and glbtag, is defined for a nonparameterized policy.

```
route-policy globalparam
  if path-type is $glbpathtype then
    set tag $glbtag
  endif
end-policy
```
When a parameterized policy has a parameter name “collision” with a global parameter name, parameters local to policy definition take precedence, effectively masking off global parameters. In addition, a validation mechanism is in place to prevent the deletion of a particular global parameter if it is referred by any policy.

**Semantics of Policy Application**

This section discusses how routing policies are evaluated and applied. The following concepts are discussed:

**Boolean Operator Precedence**

Boolean expressions are evaluated in order of operator precedence, from left to right. The highest precedence operator is NOT, followed by AND, and then OR. The following expression:

```
med eq 10 and not destination in (10.1.3.0/24) or community matches-any ([10..25]:35)
```

if fully parenthesized to display the order of evaluation, would look like this:

```
(med eq 10 and (not destination in (10.1.3.0/24))) or community matches-any ([10..25]:35)
```

The inner NOT applies only to the destination test; the AND combines the result of the NOT expression with the Multi Exit Discriminator (MED) test; and the OR combines that result with the community test. If the order of operations are rearranged:

```
not med eq 10 and destination in (10.1.3.0/24) or community matches-any ([10..25]:35)
```

then the expression, fully parenthesized, would look like the following:

```
((not med eq 10) and destination in (10.1.3.0/24)) or community matches-any ([10..25]:35)
```

**Multiple Modifications of 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
```
When Attributes Are Modified

A policy does not modify route attribute values until all tests have been completed. In other words, comparison operators always run on the initial data in the route. Intermediate modifications of the route attributes do not have a cascading effect on the evaluation of the policy. Take the following example:

```
if med eq 12 then
  set med 42
if med eq 42 then
  drop
endif
endif
```

This policy never executes the drop statement because the second test (med eq 42) sees the original, unmodified value of the MED in the route. Because the MED has to be 12 to get to the second test, the second test always returns false.
Default Drop Disposition

All route policies have a default action to drop the route under evaluation unless the route has been modified by a policy action or explicitly passed. Applied (nested) policies implement this disposition as though the applied policy were pasted into the point where it is applied.

Consider a policy to allow all routes in the 10 network and set their local preference to 200 while dropping all other routes. You might write the policy as follows:

```
route-policy two
  if destination in (10.0.0.0/8 ge 8 le 32) then
    set local-preference 200
  endif
end-policy

route-policy one
  apply two
end-policy
```

It may appear that policy one drops all routes because it neither contains an explicit `pass` statement nor modifies a route attribute. However, the applied policy does set an attribute for some routes and this disposition is passed along to policy one. The result is that policy one passes routes with destinations in network 10, and drops all others.

Control Flow

Policy statements are processed sequentially in the order in which they appear in the configuration. Policies that hierarchically reference other policy blocks are processed as if the referenced policy blocks had been directly substituted inline. For example, if the following policies are defined:

```
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
```
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.

```
routepolicy 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 routepolicy 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.

```
routepolicy 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
!`

Routing Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
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
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:

```plaintext
if med eq 8 then
    set community (12:34) additive
else
    set community (12:56) additive
endif
```

The policy language also provides syntax, using the **elseif** keyword, to string together a sequence of tests:

```plaintext
if med eq 150 then
    set local-preference 10
elseif med eq 200 then
    set local-preference 60
elseif med eq 250 then
    set local-preference 110
else
    set local-preference 0
endif
```

The statements within an **if** statement may themselves be **if** statements, as shown in the following example:

```plaintext
if community matches-any (12:34,56:78) then
    if med eq 150 then
        drop
    endif
    set local-preference 100
endif
```

This policy example sets the value of the local preference attribute to 100 on any route that has a community value of 12:34 or 56:78 associated with it. However, if any of these routes has a MED value of 150, then these routes with either the community value of 12:34 or 56:78 and a MED of 150 are dropped.

**Boolean Conditions**

In the previous section describing the **if** statement, all of the examples use simple Boolean conditions that evaluate to either true or false. RPL also provides a way to build compound conditions from simple conditions by means of Boolean operators.

Three Boolean operators exist: negation (**not**), conjunction (**and**), and disjunction (**or**). In the policy language, negation has the highest precedence, followed by conjunction, and then by disjunction. Parentheses may be used to group compound conditions to override precedence or to improve readability.

The following simple condition:

```plaintext
med eq 42
```
A simple condition may also be negated using the `not` operator:

```plaintext
not next-hop in (10.0.2.2)
```

Any Boolean condition enclosed in parentheses is itself a Boolean condition:

```plaintext
(destination in prefix-list-1)
```

A compound condition takes either of two forms. It can be a simple expression followed by the `and` operator, itself followed by a simple condition:

```plaintext
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:

```plaintext
origin is igp or origin is incomplete
```

An entire compound condition may be enclosed in parentheses:

```plaintext
(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.

```plaintext
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:

```plaintext
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:

```plaintext
(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
```

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)
```
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.

```plaintext
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:

```plaintext
route-policy NetworkControl
  if destination in (0.0.0.0/0 ge 25) then
    .
    .
```
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
    network 172.16.0.5/27 route-policy NetworkControl

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, v valid, b best
   i - internal, s stale
Origin codes: i - IGP, e - EGP, ? - incomplete

<table>
<thead>
<tr>
<th>Network</th>
<th>Next Hop</th>
<th>Metric</th>
<th>LocPrf</th>
<th>Weight</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>*&gt; 10.0.0</td>
<td>10.0.1.2</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>*&gt; 10.0.0/9</td>
<td>10.0.1.2</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>*&gt; 10.0.0/10</td>
<td>10.0.1.2</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>*&gt; 10.0.0/11</td>
<td>10.0.1.2</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>*&gt; 10.1.0.0/16</td>
<td>10.0.1.2</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>*&gt; 10.3.30.0/24</td>
<td>10.0.1.2</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>*&gt; 10.3.30.128/25</td>
<td>10.0.1.2</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>*&gt; 10.128.0.0/9</td>
<td>10.0.1.2</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>*&gt; 10.255.0.0/24</td>
<td>10.0.101.2</td>
<td>1000</td>
<td>555</td>
<td>0 100</td>
<td>e</td>
</tr>
<tr>
<td>*&gt; 10.255.64.0/24</td>
<td>10.0.101.2</td>
<td>1000</td>
<td>555</td>
<td>0 100</td>
<td>e</td>
</tr>
</tbody>
</table>
....

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:10.0.0.12
   MET ORG AS
   origin:incomplete neighbor as:3 metric:10
   aspath:3
Attributes after outbound policy was applied:
   next hop:10.0.0.12
   MET ORG AS
   origin:incomplete neighbor as:3 metric:333
   aspath:2 3
....

Table Policy

The table policy attach point allows the user to configure traffic-index values on routes as they are installed into the global routing table. This attach point supports the BGP policy accounting feature. BGP policy accounting uses the traffic indexes that are set on the BGP routes to track various counters. This way, router
operators can select different sets of BGP route attributes using the matching operations and then set different traffic indexes for each different class of route they are interested in tracking.

The following example shows how to set the traffic index to 10 in IPv4 unicast routes that originated from autonomous system 10.33. Likewise, any IPv4 unicast routes that originated from autonomous system 11.60 have their traffic index set to 11 when they are installed into the FIB. These traffic indexes are then used to count traffic being forwarded on these routes inline cards by enabling the BGP policy accounting counters on the interfaces of interest.

```
route-policy sample-table
  if as-path originates-from '10.33' then
    set traffic-index 10
  elseif as-path originates-from '11.60' then
    set traffic-index 11
  endif
end-policy

router bgp 2
  address-family ipv4 unicast
    table-policy sample-table
      .
      .
```

**Import**

The import attach point provides control over the import of routes from the global VPN IPv4 table to a particular VPN routing and forwarding (VRF) instance.

For Layer 3 VPN networks, provider edge (PE) routers learn of VPN IPv4 routes through the Multiprotocol Internal Border Gateway Protocol (MP-iBGP) from other PE routers and automatically filters out route announcements that do not contain route targets that match any import route targets of its VRFs. This automatic route filtering happens without RPL configuration; however, to provide more control over the import of routes in a VRF, you can configure a VRF import policy.

The following example shows how to perform matches based on a route target extended community and then sets the next hop. If the route has route target value 10:91, then the next hop is set to 172.16.0.1. If the route has route target value 11:92, then the next hop is set to 172.16.0.2. If the route has Site-of-Origin (SoO) value 10:111111 or 10:111222, then the route is dropped. All other non-matching routes are dropped.

```
route-policy bgpvrfr_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 bgpvrfr_import
      .
      .
```
Export

The export attach point provides control over the export of routes from a particular VRF to a global VPN IPv4 table.

For Layer 3 VPN networks, export route targets are added to the VPN IPv4 routes when VRF IPv4 routes are converted into VPN IPv4 routes and advertised through the MP-iBGP to other PE routers (or flow from one VRF to another within a PE router).

A set of export route targets is configured with the VRF without RPL configuration; however, to set route targets conditionally, you can configure a VRF export policy.

The following example shows some match and set operations supported for the export route policy. If a route matches 172.16.1.0/24 then the route target extended community is set to 10:101, and the weight is set to 211. If the route does not match 172.16.1.0/24 but the origin of the route is egp, then the local preference is set to 212 and the route target extended community is set to 10:101. If the route does not match those specified criteria, then the route target extended community 10:111222 is added to the route. In addition, RT 10:111222 is added to the route that matches any of the previous conditions as well.

route-policy bgpvrf_export
  if destination in (172.16.1.0/24) then
    set extcommunity rt (10:101)
    set weight 211
  elseif origin is egp then
    set local-preference 212
    set extcommunity rt (10:101)
  endif
  set extcommunity rt (10:111222) additive
end-policy

vrf vrf-export
  address-family ipv4 unicast
    export route-policy bgpvrf-export

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 RRIs. You can achieve scalability if you configure an RR to retain routes with a defined set of route target extended communities, and a specific set of VPNs to service.

Another reason to use this attach point is for an ASBR. ASBRs do not require that VRFs be configured, but need this configuration to retain the IPv4 VPN prefix information.

The following example shows how to configure the route policy retainer and apply it to the retain route target attach point. The route is accepted if the route contains route target extended communities 10:615, 10:6150, and 15.15.15.15.15:15. All other non-matching routes are dropped.

extcommunity-set rt rtset1
  0:615,
  10:6150,
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
```
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.

```plaintext
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.

```plaintext
as-path-set my-as-set
    ios-regex '_12$',
    ios-regex '_13$'
end-set
route-policy policy_a
    if as-path in my-as-set then
        pass
    else
        drop
    endif
end-policy
clear bgp ipv4 unicast flap-statistics route-policy policy_a
```
The debug attach point provides increased control based on prefix-based match operations. This attach point is typically used to filter debug output for various BGP commands based on the prefix of the route.

The following example shows how to configure a route policy that will only pass the prefix 20.0.0.0 with prefix length 8; therefore, the debug output shows up only for that prefix.

```plaintext
route-policy policy_b
  if destination in (10.0.0.0/8) then
    pass
  else
    drop
  endif
end-policy

debg bgp update policy_b
```
BGP Attributes and Operators

This table summarizes the BGP attributes and operators per attach points.

**Table 4: BGP Attributes and Operators**

<table>
<thead>
<tr>
<th>Attach Point</th>
<th>Attribute</th>
<th>Match</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregation</td>
<td>as-path</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>originates-from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>as-path-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>as-path-unique-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>community</td>
<td>is-empty</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td>delete in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete not in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete all</td>
</tr>
<tr>
<td></td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>extcommunity cost</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td>local-preference</td>
<td>is, ge, le, eq</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>med</td>
<td>is, eg, ge, le</td>
<td>set + set</td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>origin</td>
<td>is</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>source</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>suppress-route</td>
<td>—</td>
<td>suppress-route</td>
</tr>
<tr>
<td></td>
<td>weight</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>allocate-label</td>
<td>as-path</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>originates-from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td>as-path-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>as-path-unique-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>community</td>
<td></td>
<td>is-empty</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td></td>
</tr>
<tr>
<td>destination</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>label</td>
<td></td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td>local-preference</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>med</td>
<td></td>
<td>is, eg, ge, le</td>
<td></td>
</tr>
<tr>
<td>next-hop</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>origin</td>
<td></td>
<td>is</td>
<td></td>
</tr>
<tr>
<td>source</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>clear-policy</td>
<td>as-path</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>originates-from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td>as-path-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>as-path-unique-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>dampening</td>
<td>as-path</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>originates-from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td>as-path-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>as-path-unique-length</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>community</td>
<td></td>
<td>is-empty</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td></td>
</tr>
<tr>
<td>dampening</td>
<td></td>
<td>—/</td>
<td>set dampening</td>
</tr>
<tr>
<td>destination</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>local-preference</td>
<td></td>
<td>is, ge, le, eq</td>
<td></td>
</tr>
<tr>
<td>med</td>
<td></td>
<td>is, eg, ge, le</td>
<td></td>
</tr>
<tr>
<td>next-hop</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>origin</td>
<td></td>
<td>is</td>
<td></td>
</tr>
<tr>
<td>source</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>debug</td>
<td>destination</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>default</td>
<td>originate</td>
<td>med</td>
<td></td>
</tr>
<tr>
<td>rib-has-route</td>
<td></td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------</td>
<td>--------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>neighbor-in</td>
<td>as-path</td>
<td>in</td>
<td>prepend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local length</td>
<td>prepend most-recent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NA</td>
<td>remove as-path private-as replace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is originates-from passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>as-path-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>as-path-unique-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>community</td>
<td>is-empty</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>community with 'peeras'</td>
<td>matches-any</td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td>delete-in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete-not-in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete-all</td>
</tr>
<tr>
<td></td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>extcommunity cost</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td>extcommunity rt</td>
<td>is-empty</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td>additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td>delete-in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-within</td>
<td>delete-not-in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete-all</td>
</tr>
<tr>
<td></td>
<td>extcommunity soo</td>
<td>is-empty</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td></td>
</tr>
<tr>
<td></td>
<td>local-preference</td>
<td>is, ge, le, eq</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>med</td>
<td>is, eg, ge, le</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set +</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set -</td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------</td>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>next-hop</td>
<td>in</td>
<td>set</td>
<td>set peer address</td>
</tr>
<tr>
<td>origin</td>
<td>is</td>
<td>set</td>
<td></td>
</tr>
<tr>
<td>route-aggregated</td>
<td>route-aggregated</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>source</td>
<td>in</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weight</td>
<td>—</td>
<td>set</td>
<td></td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------</td>
<td>------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>neighbor-out</td>
<td>as-path</td>
<td>in</td>
<td>prepend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is-local</td>
<td>prepend most-recent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>length</td>
<td>remove as-path private-as private</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>replace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>neighbor-is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>originates-from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>passes-through</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique-length</td>
<td></td>
</tr>
<tr>
<td>as-path-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>as-path-unique-length</td>
<td>is, ge, le, eq</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>community</td>
<td>community with ‘peeras’</td>
<td>is-empty</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-any</td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matches-every</td>
<td>delete-in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete-not-in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>delete-all</td>
</tr>
<tr>
<td>destination</td>
<td>in</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>extcommunity cost</td>
<td>—</td>
<td>set</td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>extcommunity rt</td>
<td>is-empty</td>
<td>set</td>
<td>set additive</td>
</tr>
<tr>
<td></td>
<td>matches-any</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>matches-every</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>matches-within</td>
<td></td>
<td></td>
</tr>
<tr>
<td>extcommunity soo</td>
<td>is-empty</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td>matches-any</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>matches-every</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>matches-within</td>
<td></td>
<td></td>
</tr>
<tr>
<td>local-preference</td>
<td>is, ge, le, eq</td>
<td>set</td>
<td></td>
</tr>
<tr>
<td>med</td>
<td>is, eg, ge, le</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>-------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set +</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set -</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set max-unreachable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set igp-cost</td>
</tr>
<tr>
<td>next-hop</td>
<td>in</td>
<td></td>
<td>set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set self</td>
</tr>
<tr>
<td>origin</td>
<td>is</td>
<td></td>
<td>set</td>
</tr>
<tr>
<td>path-type</td>
<td>is</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rd</td>
<td>in</td>
<td></td>
<td></td>
</tr>
<tr>
<td>route-aggregated</td>
<td>route-aggregated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>source</td>
<td>in</td>
<td></td>
<td></td>
</tr>
<tr>
<td>unsuppress-route</td>
<td></td>
<td></td>
<td>unsuppress-route</td>
</tr>
<tr>
<td>vpn-distinguisher</td>
<td></td>
<td></td>
<td>set</td>
</tr>
<tr>
<td>neighbor-orf</td>
<td>orf-prefix</td>
<td>in</td>
<td>n/a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Implementing Routing Policy</th>
<th>BGP Attributes and Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>network</td>
<td>as-path</td>
</tr>
<tr>
<td></td>
<td>community</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>destination</td>
</tr>
<tr>
<td></td>
<td>extcommunity cost</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mpls-label</td>
</tr>
<tr>
<td></td>
<td>local-preference</td>
</tr>
<tr>
<td></td>
<td>med</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
</tr>
<tr>
<td></td>
<td>origin</td>
</tr>
<tr>
<td></td>
<td>route-type</td>
</tr>
<tr>
<td></td>
<td>tag</td>
</tr>
<tr>
<td></td>
<td>weight</td>
</tr>
<tr>
<td>next-hop</td>
<td>destination</td>
</tr>
<tr>
<td></td>
<td>protocol</td>
</tr>
<tr>
<td></td>
<td>source</td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>redistribute</td>
<td>as-path</td>
</tr>
<tr>
<td></td>
<td>community</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>destination</td>
<td>in</td>
</tr>
<tr>
<td>extcommunity cost</td>
<td>—</td>
</tr>
<tr>
<td>local-preference</td>
<td>—</td>
</tr>
<tr>
<td>med</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>next-hop</td>
<td>in</td>
</tr>
<tr>
<td>origin</td>
<td>—</td>
</tr>
<tr>
<td>mpls-label</td>
<td>route-has-label</td>
</tr>
<tr>
<td>route-type</td>
<td>is</td>
</tr>
<tr>
<td>tag</td>
<td>is, eq, ge, le</td>
</tr>
<tr>
<td>weight</td>
<td>—</td>
</tr>
<tr>
<td>retain-rt</td>
<td>extcommunity rt</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Implementing Routing Policy

Routing Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
Some BGP route attributes are inaccessible from some BGP attach points for various reasons. For example, the **set med **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 5: Restricted BGP Operations by Attach Point**

<table>
<thead>
<tr>
<th>Command</th>
<th>import</th>
<th>export</th>
<th>aggregation</th>
<th>redistribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>prepend as-path most-recent</td>
<td>eBGP only</td>
<td>eBGP only</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>replace as-path</td>
<td>eBGP only</td>
<td>eBGP only</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>set med igp-cost</td>
<td>forbidden</td>
<td>eBGP only</td>
<td>forbidden</td>
<td>forbidden</td>
</tr>
<tr>
<td>set weight</td>
<td>n/a</td>
<td>forbidden</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>suppress</td>
<td>forbidden</td>
<td>forbidden</td>
<td>n/a</td>
<td>forbidden</td>
</tr>
</tbody>
</table>

**Default-Information Originate**

The default-information originate attach point allows the user to conditionally inject the default route 0.0.0.0/0 into the OSPF link-state database, which is done by evaluating the attached policy. If any routes in the local RIB pass the policy, then the default route is inserted into the link-state database.

The following example shows how to generate a default route if any of the routes that match 10.0.0.0/8 ge 8 le 25 are present in the RIB:

```plaintext
route-policy ospf-originate
  if rib-has-route in (10.0.0.0/8 ge 8 le 25) then
    pass
  endif
end-policy

router ospf 1
  default-information originate policy ospf-originate
  .
  .
```

**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 cost 100
  elseif tag eq 20 then
    set ospf cost 200
  else
    drop
  endif
end-policy
router ospf 1
  redistribute isis instance_10 policy OSPF-redist
```

### Area-in

The area-in attach point within OSPF allows you to filter inbound OSPF type-3 summary link-state advertisements (LSAs). The attach point provides prefix-based matching and hence increased control for filtering type-3 summary LSAs.

The following example shows how to configure the prefix for OSPF summary LSAs. If the prefix matches any of 10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24, it is accepted. If the prefix matches any of 10.106.3.0/24, 10.106.7.0/24, 10.106.13.0/24, it is dropped.

```
route-policy OSPF-area-in
  if destination in (10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24) then
    drop
  endif
  if destination in (10.106.3.0/24, 10.106.7.0/24, 10.106.13.0/24) then
    pass
  endif
end-policy
router ospf 1
  area 1
    route-policy OSPF-area-in in
```

### Area-out

The area-out attach point within OSPF allows you to filter outbound OSPF type-3 summary LSAs. The attach point provides prefix-based matching and, hence, increased control for filtering type-3 summary LSAs.

The following example shows how to configure the prefix for OSPF summary LSAs. If the prefix matches any of 10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24, it is announced. If the prefix matches any of 10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24, it is dropped and not announced.

```
route-policy OSPF-area-out
```
if destination in (10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24) then
drop
endif
if destination in (10.105.3.0/24, 10.105.7.0/24, 10.105.13.0/24) then
pass
endif
end-policy
router ospf 1
area 1
route-policy OSPF-area-out out

### OSPF Attributes and Operators

This table summarizes the OSPF attributes and operators per attach points.

**Table 6: OSPF Attributes and Operators**

<table>
<thead>
<tr>
<th>Attach Point</th>
<th>Attribute</th>
<th>Match</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>default-information originate</td>
<td>ospf-metric</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>metric-type</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>rib-has-route</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>redistribute</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>metric-type</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>ospf-metric</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>next-hop</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>mpls-label</td>
<td>route-has-label</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>rib-metric</td>
<td>is, le, ge, eq</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>route-type</td>
<td>is</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>is, eq, ge, le</td>
<td>set</td>
</tr>
<tr>
<td>area-in</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>area-out</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>Attach Point</td>
<td>Attribute</td>
<td>Match</td>
<td>Set</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>-------</td>
<td>-----</td>
</tr>
<tr>
<td>spf-prefix-priority</td>
<td>destination</td>
<td>in</td>
<td>n/a</td>
</tr>
<tr>
<td>spf-priority</td>
<td>n/a</td>
<td>set</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td>is, le, ge, eq</td>
<td></td>
<td>n/a</td>
</tr>
</tbody>
</table>

**Distribute-list in**

The distribute-list in attach point within OSPF allows use of route policies to filter OSPF prefixes. The distribute-list in route-policy can be configured at OSPF instance, area, and interface levels. The route-policy used in the distribute-list in command supports match statements, "destination" and "rib-metric". The "set" commands are not supported in the route-policy.

These are examples of valid route-policies for "distribute-list in":

```plaintext
route-policy DEST
  if destination in (10.10.10.10/32) then
    drop
  else
    pass
  endif
end-policy

route-policy METRIC
  if rib-metric ge 10 and rib-metric le 19 then
    drop
  else
    pass
  endif
end-policy

prefix-set R-PFX
  10.10.10.30
end-set

route-policy R-SET
  if destination in R-PFX and rib-metric le 20 then
    pass
  else
    drop
  endif
end-policy
```

**OSPFv3 Policy Attach Points**

This section describes each of the OSPFv3 policy attach points and provides a summary of the OSPFv3 attributes and operators.

**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 7: OSPFv3 Attributes and Operators**

<table>
<thead>
<tr>
<th>Attach Point</th>
<th>Attribute</th>
<th>Match</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ospf-metric</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>metric-type</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>rib-has-route</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td>redistribute</td>
<td>destination</td>
<td>in</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>ospf-metric</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>metric-type</td>
<td>—</td>
<td>set</td>
</tr>
<tr>
<td></td>
<td>route-type</td>
<td>is</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>is, eq, ge, le</td>
<td>—</td>
</tr>
</tbody>
</table>
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 metric 100
    set level level-1-2
  endif
end-policy

router isis instance_10
  address-family ipv4 unicast
    default-information originate
      policy isis_originate
```

Inter-area-propagate

The inter-area-propagate attach point within IS-IS allows the prefixes to be conditionally propagated from one level to another level within the same IS-IS instance.

The following example shows how to allow prefixes to be leaked from the level 1 LSP into the level 2 LSP if any of the prefixes match 10.0.0.0/8 ge 8 le 25.

```
route-policy isis-propagate
  if destination in (10.0.0.0/8 ge 8 le 25) then
    pass
  endif
end-policy

router isis instance_10
  address-family ipv4 unicast
    propagate level 1 into level 2 policy isis-propagate
```

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
```
Nondestructive Editing of Routing Policy

The Nondestructive Editing of Routing Policy changes the default exit behavior under routing policy configuration mode to abort the configuration.

The default exit command acts as end-policy, end-set, or end-if. If the exit command is executed under route policy configuration mode, the changes are applied and configuration is updated. This destructs the existing policy. The rpl set-exit-as-abort command allows to overwrite the default behavior of the exit command under the route policy configuration mode.

Attached Policy Modification

Policies that are in use do, on occasion, need to be modified. In the traditional configuration model, a policy modification would be done by completely removing the policy and 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.

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
- Vim

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:

```plaintext
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*:

```plaintext
route-policy Child A
  if destination in (10.10.0.0/16) then
    set local-pref 111
  endif
end-policy
!

route-policy Child B
  if as-path originates-from '222' then
    set community (333:222) additive
  endif
end-policy
!

route-policy Parent
  if apply Child A and apply Child B then
    set community (333:333) additive
  else
    set community (333:444) additive
  endif
end-policy
!
```

In the above scenarios, whenever the policy *Parent* is executed, the decision of the "if" condition in that is selected based on the result of policies *Child A* and *Child B*. The policy *Parent* is equivalent to policy *merged* as given below:

```plaintext
route-policy merged
```
if destination in (10.10.0.0/16) and as-path originates-from '222' then
  set local-pref 111
  set community (333:222, 333:333) additive
elseif destination in (10.10.0.0/16) then /*Only Policy Child A is pass */
  set local-pref 111
  set community (333:222, 333:444) additive /*From else block */
elseif as-path originates-from '222' then /*Only Policy Child B is pass */
  set community (333:222, 333:444) additive /*From else block */
else
  set community (333:444) additive /*From else block */
endif
end-policy

Apply Conditions can be used with parameters and are supported on all attach points and on all clients. Hierarchical Apply Conditions can be used without any constraints on a cascaded level.

Existing route policy semantics can be expanded to include this Apply Condition:

Route-policy policy_name
  If apply policyA and apply policyB then
    Set med 100
  Else if not apply policyD then
    Set med 200
  Else
    Set med 300
  Endif
End-policy

Behavior of pass/drop/done RPL Statements for Simple Hierarchical Policies

This table describes the behavior of pass/drop/done RPL statements, with a possible sequence for executing the done statement for Simple Hierarchical Policies.

<table>
<thead>
<tr>
<th>Route-policies with simple hierarchical policies</th>
<th>Possible done statement execution sequence</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>pass</td>
<td>pass</td>
<td>Marks the prefix as &quot;acceptable&quot; and continues with execution of continue_list statements.</td>
</tr>
<tr>
<td></td>
<td>Continue_list</td>
<td></td>
</tr>
<tr>
<td>drop</td>
<td>Stmts_list</td>
<td>ReJECTS the route immediately on hitting the drop statement and stops policy execution.</td>
</tr>
<tr>
<td></td>
<td>drop</td>
<td></td>
</tr>
<tr>
<td>done</td>
<td>Stmts_list</td>
<td>Accepts the route immediately on hitting the done statement and stops policy execution.</td>
</tr>
<tr>
<td></td>
<td>done</td>
<td></td>
</tr>
<tr>
<td>pass followed by done</td>
<td>pass</td>
<td>Exits immediately at the done statement with &quot;accept route&quot;.</td>
</tr>
<tr>
<td></td>
<td>Statement_list</td>
<td></td>
</tr>
<tr>
<td></td>
<td>done</td>
<td></td>
</tr>
</tbody>
</table>
Behavior of pass/drop/done RPL Statements for Hierarchical Policy Conditions

This section describes the behavior of **pass**/**drop**/**done** RPL statements, with a possible sequence for executing the **done** statement for Hierarchical Policy Conditions.

Terminology for policy execution: "true-path", "false-path", and "continue-path".

<table>
<thead>
<tr>
<th>Route-policies with simple hierarchical policies</th>
<th>Possible done statement execution sequence</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>drop</strong> followed by <strong>done</strong></td>
<td><strong>drop</strong></td>
<td>This is an invalid scenario at execution point of time. Policy terminates execution at the <strong>drop</strong> statement itself, without going through the statement list or the <strong>done</strong> statement; the prefix will be rejected or dropped.</td>
</tr>
</tbody>
</table>

### Behavior of pass/drop/done RPL Statements for Hierarchical Policy Conditions

This section describes the behavior of **pass**/**drop**/**done** RPL statements, with a possible sequence for executing the **done** statement for Hierarchical Policy Conditions.

Terminology for policy execution: "true-path", "false-path", and "continue-path".

**Route-policy parent**

- **If apply hierarchical_policy_condition then**
  - TRUE-PATH : if hierarchical_policy_condition returns TRUE then this path will be executed.
  - Else
    - FALSE-PATH : if hierarchical_policy_condition returns FALSE then this path will be executed.
  - End-if
- CONTINUE-PATH : Irrespective of the TRUE/FALSE this path will be executed.
- End-policy

<table>
<thead>
<tr>
<th>Hierarchical policy conditions</th>
<th>Possible done statement execution sequence</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pass</strong></td>
<td><strong>pass</strong></td>
<td>Marks the return value as &quot;true&quot; and continues execution within the same policy condition. If there is no statement after &quot;<strong>pass</strong>&quot;, returns &quot;true&quot;.</td>
</tr>
<tr>
<td><strong>pass</strong> followed by <strong>done</strong></td>
<td><strong>pass</strong> or set action statement</td>
<td>Marks the return value as &quot;true&quot; and continues execution till the <strong>done</strong> statement. Returns &quot;true&quot; to the apply policy condition to take &quot;true-path&quot;.</td>
</tr>
<tr>
<td><strong>done</strong></td>
<td>Stmt_list without <strong>pass</strong> or set</td>
<td>Returns &quot;false&quot;. Condition takes &quot;false-path&quot;.</td>
</tr>
<tr>
<td><strong>drop</strong></td>
<td>Stmt_list</td>
<td>The prefix is dropped or rejected.</td>
</tr>
</tbody>
</table>
Nested Wildcard Apply Policy

The hierarchical constructs of Routing Policy Language (RPL) allows one policy to refer to another policy. The referred or called policy is known as a child policy. The policy from which another policy is referred is called calling or parent policy. A calling or parent policy can nest multiple child policies for attachment to a common set of BGP neighbors. The nested wildcard apply policy allows wildcard (*) based apply nesting. The wildcard operation permits declaration of a generic apply statement that calls all policies that contain a specific defined set of alphanumeric characters, defined on the router.

A wildcard is specified by placing an asterisk (*) at the end of the policy name in an apply statement. Passing parameters to wildcard policy is not supported. The wildcard indicates that any value for that portion of the apply policy matches.

To illustrate nested wildcard apply policy, consider this policy hierarchy:

```plaintext
route-policy Nested_Wilcard
apply service_policy_customer*
end-policy

route-policy service_policy_customer_a
if destination in prfx_set_customer_a then
set extcommunity rt (1:1) additive
endif
end-policy

route-policy service_policy_customer_b
if destination in prfx_set_customer_b then
set extcommunity rt (1:1) additive
endif
end-policy

route-policy service_policy_customer_c
if destination in prfx_set_customer_c then
set extcommunity rt (1:1) additive
endif
end-policy
```

Here, a single parent apply statement (apply service_policy_customer*) calls (inherits) all child policies that contain the identified character string "service_policy_customer". As each child policy is defined globally, the parent dynamically nests the child policies based on the policy name. The parent is configured once and inherits each child policy on demand. There is no direct association between the parent and the child policies beyond the wildcard match statement.

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 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 202 provides additional conceptual information on static routes.

This module describes how to implement static routes.

- Restrictions for Implementing Static Routes, on page 193
- Configure Static Route, on page 194
- Floating Static Routes, on page 195
- Configure Static Routes Between PE-CE Routers, on page 197
- IPv4 Multicast Static Routes, on page 198
- Default VRF, on page 200
- Configure Native UCMP for Static Routing, on page 201
- References for Static Routes, on page 202

Restrictions for Implementing Static Routes

These restrictions apply while implementing Static Routes:

- Currently, only default VRF is supported. L3VPN, VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families will be supported in a future release.

- 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.

Note
Currently, only default VRF is supported. VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families will be supported in a future release.

This task explains how to configure a static route.

SUMMARY STEPS

1. configure
2. router static
3. vrf vrf-name
4. address-family { ipv4 | ipv6 } { unicast | multicast }
5. prefix mask [vrf vrf-name] [ ip-address | interface-type interface-instance ] [ distance ] [ description text ] [ tag tag ] [ permanent ]
6. commit

DETAILED STEPS

Step 1 configure
Step 2 router static
Example:

```
RP/0/RP0/CPU0:router(config)# router static
```
Enters static route configuration mode.

Step 3 vrf vrf-name
Example:

```
RP/0/RP0/CPU0:router(config-static)# vrf vrf_A
```
(Optional) Enters VRF configuration mode.
If a VRF is not specified, the static route is configured under the default VRF.

Step 4 address-family { ipv4 | ipv6 } { unicast | multicast }
Example:

```
RP/0/RP0/CPU0:router(config-static-vrf)# address family ipv4 unicast
```
Enters address family mode.

**Step 5**

```
prefix mask [vrf vrf-name] { ip-address | interface-type interface-instance } [ distance ] [ description text ] [ tag tag ] [ permanent ]
```

**Example:**

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 10.0.0.0/8 172.20.16.6 110
```

Configures an administrative distance of 110.

- This example shows how to route packets for network 10.0.0.0 through to a next hop at 172.20.16.6 if dynamic information with administrative distance less than 110 is not available.

**Step 6**

```
commit
```

A default static route is often used in simple router topologies. In the following example, a route is configured with an administrative distance of 110.

```
configure
router static
    address-family ipv4 unicast
    0.0.0.0/0 2.6.0.1 110
end
```

---

**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.

**SUMMARY STEPS**

1. configure
2. router static
3. vrf vrf-name
4. address-family { ipv4 | ipv6 } { unicast | multicast }
5. `prefix mask [vrf vrf-name ] { ip-address | interface-type interface-instance } [ distance ] [ description text ] [ tag tag ] [ permanent ]`

6. `commit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>configure</code></td>
</tr>
<tr>
<td>2</td>
<td><code>router static</code></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config)# router static</code></td>
</tr>
<tr>
<td></td>
<td>Enters static route configuration mode.</td>
</tr>
<tr>
<td>3</td>
<td><code>vrf vrf-name</code></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-static)# vrf vrf_A</code></td>
</tr>
<tr>
<td></td>
<td>(Optional) Enters VRF configuration mode.</td>
</tr>
<tr>
<td></td>
<td>If a VRF is not specified, the static route is configured under the default VRF.</td>
</tr>
<tr>
<td>4</td>
<td>`address-family { ipv4</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-static-vrf)# address family ipv6 unicast</code></td>
</tr>
<tr>
<td></td>
<td>Enters address family mode.</td>
</tr>
<tr>
<td>5</td>
<td>`prefix mask [vrf vrf-name ] { ip-address</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201</code></td>
</tr>
<tr>
<td></td>
<td>Configures an administrative distance of 201.</td>
</tr>
<tr>
<td>6</td>
<td><code>commit</code></td>
</tr>
</tbody>
</table>

A floating static route is often used to provide a backup path if connectivity fails. In the following example, a route is configured with an administrative distance of 201.

```
configure
router static
address-family ipv6 unicast
  2001:0DB8::/32 2001:0DB8:3000::1 201
end
```
Configure Static Routes Between PE-CE Routers

This task explains how to configure static routing between PE-CE routers.

**Note**

VRF fallback is not supported with IPv6 VPN Provider Edge (6VPE).

### SUMMARY STEPS

1. configure
2. router static
3. vrf vrf-name
4. address-family { ipv4 | ipv6 } { unicast | multicast }
5. prefix mask {vrf vrf-name} { ip-address | interface-type interface-path-id } [ distance ] [ description text ] [ tag tag ] [ permanent ]
6. commit

### DETAILED STEPS

**Step 1** configure

**Step 2** router static

**Example:**

RP/0/RP0/CPU0:router(config)# router static

Enters static route configuration mode.

**Step 3** vrf vrf-name

**Example:**

RP/0/RP0/CPU0:router(config-static)# vrf vrf_A

(Optional) Enters VRF configuration mode.

If a VRF is not specified, the static route is configured under the default VRF.

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

**Example:**

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

Enters address family mode.

**Step 5** prefix mask {vrf vrf-name} { ip-address | interface-type interface-path-id } [ distance ] [ description text ] [ tag tag ] [ permanent ]

**Example:**
In the following example, a static route between PE and CE routers is configured, and a VRF is associated with the static route:

```plaintext
configure
gerouter 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 5: 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:

```plaintext
/* 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 0/0/0/12
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 0/0/0/12
    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*
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.

**SUMMARY STEPS**

1. configure
2. router static
3. vrf vrf-name
4. address-family { ipv4 | ipv6 } { unicast | multicast }
5. prefix mask [vrf vrf-name] [next-hop ip-address | interface-name] {path-id} [distance] [description text] [tag tag] [permanent]
6. commit

**DETAILED STEPS**

**Step 1**
configure

**Step 2**
router static

**Example:**
```
RP/0/
/CPU0:router(config)# router static
```
Enter static route configuration mode.

**Step 3**
vrf vrf-name

**Example:**
Configure Native UCMP for Static Routing

In a network where traffic is load balanced on two or more links, configuring equal metrics on the links would create Equal Cost Multipath (ECMP) next hops. Because the bandwidth of the links is not taken into consideration while load balancing, the higher bandwidth links are underutilized. To avoid this problem, you can configure Unequal Cost Multipath (UCMP), either locally (local UCMP), or natively (native UCMP) so that the higher bandwidth links carry traffic in proportion to the capacity of the links. UCMP supports IPv4 and IPv6 static VRF routes.

**Local UCMP**: All static routes are configured with the same link metrics. The static IGP calculates the load metric based on the bandwidth of the links and load balances the traffic across the links. However, local UCMP does not consider bandwidth while load balancing across links that are closer to the destination (multiple hops away).

**Native UCMP**: Static routes over higher bandwidth links are configured with lower link metrics so that they are preferred to routes over lower bandwidth links. The static IGP calculates the load metric based on the bandwidth of the links and determines the percentage of traffic going out of the higher and lower bandwidth links. By matching the configured link metrics with end-to-end available bandwidth, native UCMP is able to effectively load balance traffic across links that are closer to the destination (multiple hops away).

**Configuration Example**

Consider the topology in the following figure. For load balancing traffic out of Router A1, if local UCMP is used, then both 10G and 100G links will have equal link metrics. The static IGP decides to send more traffic out of the 100G link because of the higher load metric. However, for load balancing traffic out of Router A2, local UCMP works only on links to Routers C1 and C2. For load balancing traffic from Router C1 to Router A1 and Router C2 to Router A1, native UCMP is preferred. As a result, local UCMP is used only on single hop destinations, and native UCMP is used for multi-hop destinations.
To configure UCMP for static routing, use the following steps:

1. Enter the global configuration mode.
   ```
   RP/0/0/CPU0:Router# configure
   ```

2. Enter the static routing mode.
   ```
   RP/0/0/CPU0:Router(config)# router static
   ```

3. Configure UCMP with load metric for IPv4 or IPv6 address family.
   ```
   RP/0/0/CPU0:Router(config-static)# address-family ipv4 unicast
   RP/0/0/CPU0:Router(config-static-afi)# 10.10.10.1/32 GigabitEthernet 0/0/0/1 metric 10
   ```

   In this example, we have configured UCMP for IPv4 address family. To configure UCMP for IPv6 address family, use the following sample configuration.
   ```
   RP/0/0/CPU0:Router(config-static)# address-family ipv6 unicast
   RP/0/0/CPU0:Router(config-static-afi)# 10::1/64 GigabitEthernet 0/0/0/1 metric 10
   ```

4. Exit the static configuration mode and commit your configuration.
   ```
   RP/0/0/CPU0:Router(config-static-afi)# exit
   RP/0/0/CPU0:Router(config-static)# exit
   RP/0/0/CPU0:Router(config)# commit
   ```

   Repeat this procedure on all routers that need to be configured with UCMP.

---

**References for Static Routes**

The following topics provide additional conceptual information on static routes:

- Static Route Functional Overview, on page 203
- Default Administrative Distance, on page 203
- Directly Connected Routes, on page 203
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-safi)# 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.

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::/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 7

Implementing BFD

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

- BFD Overview, on page 207
- BFD over Bundle, on page 212
- BFD Transparency, on page 216
- Bidirectional Forwarding Detection over Logical Bundle, on page 221
- IPv4 Multihop BFD, on page 226

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.

Restrictions

These restrictions apply to BFD:

- Demand mode is not supported in Cisco IOS XR software.
- BFD echo mode and encryption are not supported.
- BFD hardware offload for IPv4 is supported.
- Only the static, OSPF, BGP and IS-IS applications are supported on BFD.
- BFD dampening for IPv4 is supported starting from Cisco IOS XR Release 6.3.2.
- BFD multihop is supported starting from IOS XR Release 6.3.2 but BFD multihop over non IP core is not supported.
- Only static routes are supported in IPv6 BFD.
- BFD is only supported in IP core. It cannot coexist with Label distribution Protocol, or Segment Routing, or Traffic Engineering in the core.
Enabling BFD on a Static Route

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

RP/0/RP0/CPU0:router(config)# configure
/*Enter static route configuration mode, and configure static routing. */
RP/0/RP0/CPU0:router(config)# router static
/*Enter address family configuration mode. */
RP/0/RP0/CPU0:router(config-static)# address-family ipv4 unicast 192.168.2.2/32
/*Specify an unicast destination address and next-hop IPv4 address. Enable BFD fast-detection on the specified IPv4 unicast destination address */
RP/0/RP0/CPU0:router(config-static)# 192.168.2.2 192.168.6.2 bfd fast-detect minimum-interval 4 multiplier 3

Running Configuration
router static
    address-family ipv4 unicast
        192.168.2.2 192.168.6.2 bfd fast-detect minimum-interval 4 multiplier 3
!
!

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.

SUMMARY STEPS

1. configure
2. router ospf process-name
3. area area-id
4. interface type interface-path-id
5. bfd fast-detect
6. bfd minimum-interval milliseconds
7. bfd multiplier multiplier
8. commit
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
<td></td>
</tr>
</tbody>
</table>
| **Step 2** | router ospf *process-name*  
Example:  
RP/0/RP0/CPU0:router(config)# router ospf 0 | Enters OSPF configuration mode, allowing you to configure the OSPF routing process.  
*Note*: 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** | commit |  |

### Enable BFD for IS-IS on an Interface

Perform the following steps to configure BFD for Integrated Intermediate System-to-Intermediate System (IS-IS) on an interface.

*Note*: BFD per interface configuration is supported for OSPF and IS-IS only.
Router# configure

/* Enter IS-IS configuration mode to configure the IS-IS routing process. */
Router(config)# router isis 65444

/* Set the system type (area or backbone router). Each IS-IS instance can support either a
single Level 1 or Level 2, or one of each. */
Router(config-isis)# is-type level-2-only

/* Specify a NET for each routing instance if you are configuring multi-instance IS-IS. */
Router(config-isis)# net 49.0001.0840.3803.4088.00

/* Enter interface configuration mode. */
Router(config-isis)# interface gigabitEthernet 0/3/0/1

/* Set the BFD minimum interval. */
Router(config-isis-if)# bfd minimum-interval 6500

/* Set the BFD multiplier. */
Router(config-isis-if)# bfd multiplier 7

/* Enable BFD to detect failures in the path between adjacent forwarding engines. Only IPv4
is supported. */
Router(config-isis-if)# bfd fast-detect ipv4

/* Specify the IPv4 address family and enters router address family configuration mode. */
Router(config-isis-if)# address-family ipv4 unicast

Running Configuration

configure
    router isis 65444
        is-type level-2-only
        net 49.0001.0840.3803.4088.00
        interface gigabitEthernet 0/3/0/1
        bfd minimum-interval 6500
        bfd multiplier 7
        bfd fast-detect ipv4
        address-family ipv4 unicast

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.

SUMMARY STEPS

1. configure
2. router bgp autonomous-system-number
3. neighbor ip-address
4. remote-as autonomous-system-number
5. bfd fast-detect
6. bfd minimum-interval milliseconds
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>configure</td>
<td>Enters BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Step 2</td>
<td>router bgp autonomous-system-number</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>neighbor ip-address</td>
<td>Creates a neighbor and assigns it a remote autonomous system. This example configures the remote autonomous system to be 2002.</td>
</tr>
<tr>
<td></td>
<td>remote-as autonomous-system-number</td>
<td>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 networking devices and the neighbor 172.168.40.24.</td>
</tr>
<tr>
<td></td>
<td>bfd fast-detect</td>
<td>Sets the BFD minimum interval. Range is 4-30000 milliseconds.</td>
</tr>
<tr>
<td></td>
<td>bfd minimum-interval milliseconds</td>
<td>Sets the BFD multiplier. This is optional, the minimum is 3 and by default the multiplier will be 3 for all protocols</td>
</tr>
<tr>
<td></td>
<td>bfd multiplier multiplier</td>
<td><strong>commit</strong></td>
</tr>
</tbody>
</table>
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/RSP0/CPU0:router# show bfd counters all packet location 0/3/cpu0
RP/0/RSP0/CPU0:router# clear bfd counters all packet location 0/3/cpu0
RP/0/RSP0/CPU0:router# show bfd counters all packet location 0/3/cpu0
```

BFD over Bundle

BFD over Bundle (BoB) mode is a standard based fast failure detection of link aggregation (LAG) member links that is interoperable between different platforms. BoB on a per bundle basis supports only IETF standard per bundle.

Note

- For BFD over Bundle, the BFD client is bundlemgr. Hence if BFD session goes down, bundlemgr will bring down the bundle, and this in turn brings down the routing session.
- In a BFD over Bundle scenario, having the BFD secondary interface on the same line card might result in traffic loss and adjacency loss during an LC reload.

Restrictions

To support BFD on bundle member links, ensure that the routers on either end of the bundle are connected back-to-back without a Layer 2 switch in between.

Enabling BFD Sessions on Bundle Members

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

**SUMMARY STEPS**

1. configure
2. interface Bundle-Ether `bundle-id`
3. bfd address-family ipv4 fast-detect
4. bfd mode ietf
5. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 interface Bundle-Ether <code>bundle-id</code></td>
<td>Enters interface configuration mode for the specified bundle ID.</td>
</tr>
</tbody>
</table>
### Implementing BFD

#### Specifying the BFD Destination Address on a Bundle

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

**SUMMARY STEPS**

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

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> interface Bundle-Ether bundle-id</td>
<td>Enters interface configuration mode for the specified bundle ID.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> bfd address-family ipv4 destination ip-address</td>
<td>Specifies the primary IPv4 address assigned to the bundle interface on a connected remote system, where <code>ip-address</code> is the 32-bit IP address in dotted-decimal format (A.B.C.D).</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-if)# bfd address-family ipv4 destination 10.20.20.1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>
Configuring the Minimum Thresholds for Maintaining an Active Bundle

The bundle manager uses two configurable minimum thresholds to determine whether a bundle can be brought up or remain up, or is down, based on the state of its member links:

- Minimum active number of links
- Minimum active bandwidth available

Whenever the state of a member changes, the bundle manager determines whether the number of active members or available bandwidth is less than the minimum. If so, then the bundle is placed, or remains, in DOWN state. Once the number of active links or available bandwidth reaches one of the minimum thresholds, then the bundle returns to the UP state.

To configure minimum bundle thresholds, complete these steps:

**SUMMARY STEPS**

1. configure
2. interface Bundle-Ether bundle-id
3. bundle minimum-active bandwidth kbps
4. bundle minimum-active links links
5. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 interface Bundle-Ether bundle-id</td>
<td>Enters interface configuration mode for the specified bundle ID.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1</td>
</tr>
<tr>
<td>Step 3 bundle minimum-active bandwidth kbps</td>
<td>Sets the minimum amount of bandwidth required before a bundle can be brought up or remain up. The range is from 1 through a number that varies depending on the platform and the bundle type.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000</td>
</tr>
<tr>
<td>Step 4 bundle minimum-active links links</td>
<td>Sets the number of active links required before a bundle can be brought up or remain up. The range is from 1 to 32.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-if)# bundle minimum-active links 2</td>
</tr>
<tr>
<td>Step 5 commit</td>
<td></td>
</tr>
</tbody>
</table>

**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.
## 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:

### SUMMARY STEPS

1. configure
2. interface Bundle-Ether \textit{bundle-id}
3. bfd address-family ipv4 minimum-interval \textit{milliseconds}
4. bfd address-family ipv4 multiplier \textit{multiplier}
5. commit

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td>Enters interface configuration mode for the specified bundle ID.</td>
</tr>
<tr>
<td>Step 2</td>
<td>interface Bundle-Ether \textit{bundle-id}</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>bfd address-family ipv4 minimum-interval \textit{milliseconds}</td>
<td>Specifies the minimum interval, in milliseconds, for asynchronous mode control packets on IPv4 BFD sessions on bundle member links. The range is from 4 to 30000.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-if)# bfd address-family ipv4 minimum-interval 2000</td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td></td>
<td>Although the command allows you to configure a minimum of 2, the supported minimum is 3.</td>
</tr>
<tr>
<td>Step 4</td>
<td>bfd address-family ipv4 multiplier \textit{multiplier}</td>
<td>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.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-if)# bfd address-family ipv4 multiplier 30</td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td></td>
<td>Although the command allows you to configure a minimum of 2, the supported minimum is 3.</td>
</tr>
</tbody>
</table>
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:

**SUMMARY STEPS**

1. `configure`
2. `interface Bundle-Ether bundle-id`
3. `bfd mode ietf`
4. `bfd address-family ipv4 fast-detect`
5. `commit`
6. `show bundle bundle-ether bundle-id`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>configure</code></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>interface Bundle-Ether bundle-id</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 1</code></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><code>bfd mode ietf</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RP0/CPU0:router(config-if)# bfd mode ietf</code></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><code>bfd address-family ipv4 fast-detect</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RP0/CPU0:router(config-if)# bfd address-family ipv4 fast-detect</code></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><code>commit</code></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><code>show bundle bundle-ether bundle-id</code></td>
</tr>
</tbody>
</table>

**BFD Transparency**

Bidirectional Forwarding Detection (BFD) protocol is a simple hello mechanism that detects failures in a network in less than one second, depending on the timer value configured.
Both endpoints of a BFD Session periodically send Hello packets to each other. If a number of those packets are not received, the session is considered down. BFD provides fast BFD peer failure detection times independently of all media types, encapsulations, topologies, and routing protocols BGP, IS-IS, and OSPF.

BFD Transparency feature enables you to configure BFD Sessions between customer edge devices connected over an L2VPN network. These BFD sessions are transparent to the core. For example, BFD packets being exchanged between CEs are neither dropped on any router in the core, nor punted on any core device.

In this section, you will learn how to configure BFD Transparency in Ethernet VPN (EVPN) Virtual Private Wire Service (VPWS).

**Ethernet VPN Virtual Private Wire Service**

EVPN VPWS (Ethernet VPN Virtual Private Wire Service) is a BGP control plane solution for point-to-point services. It implements signaling and encapsulation techniques for establishing an EVPN instance between a pair of provider edge devices.

EVPN VPWS supports both single-homing and multi-homing.

**Configuration**

The following sections describes the procedure for configuring IP Fast Reroute with Remote LFA.

- Configure L2VPN on the provide edge router
- Configure BFD on the customer edge router

**Configure L2VPN on the Provide Edge Router**

Configure L2VPN on the provider edge router.

```plaintext
/* Enable IS-IS and configure routing level for an area. */
RP/0/CPU0:router# configure
RP/0/CPU0:router(config)# interface tengige 0/1/0/8/2.1
RP/0/CPU0:router(config-subif)# exit
RP/0/CPU0:router(config)# router isis
RP/0/CPU0:router(config-isis)# is-type level-2-only
RP/0/CPU0:router(config-isis)# net 49.1234.2222.2222.2222.00
RP/0/CPU0:router(config-isis)# nsr
RP/0/CPU0:router(config-isis)# nsf cisco
RP/0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/CPU0:router(config-isis-af)# metric style wide
RP/0/CPU0:router(config-isis-af)# end
RP/0/CPU0:router(config)# interface Bundle-Ether 199
RP/0/CPU0:router(config-if)# address-family ipv4 unicast
RP/0/CPU0:router(config-if)# end
RP/0/CPU0:router(config)# interface Loopback 0
RP/0/CPU0:router(config-if)# end
RP/0/CPU0:router(config-if)# address-family ipv4 unicast
RP/0/CPU0:router(config-if)# exit

/* Configure L2VPN EVPN address family. */
RP/0/CPU0:router(config)# router bgp 100
RP/0/CPU0:router(config-bgp)# bpg router-id 10.10.10.1
RP/0/CPU0:router(config-bgp)# address-family 12vpn evpn
RP/0/CPU0:router(config-bgp)# neighbor 192.0.2.1
RP/0/CPU0:router(config-bgp-nbr)# remote-as 100
```
/* Configure MPLS LDP for the physical core interface. */
RP/0//CPU0:router(config-bgp-nbr-af)# mpls ldp
RP/0//CPU0:router(config-bgp-nbr-af)# exit
RP/0//CPU0:router(config-bgp)# interface Bundle-Ether 199
RP/0//CPU0:router(config-if)# exit

/* Configure L2VPN Xconnect. */
RP/0//CPU0:router(config)# 12vpn
RP/0//CPU0:router(config-12vpn)# router-id 10.10.10.1
RP/0//CPU0:router(config-12vpn)# xconnect group bfdtr
RP/0//CPU0:router(config-12vpn-xf)# p2p vpws-ce
RP/0//CPU0:router(config-12vpn-xf)# interface TenGigE 0/4/0/2/0.1
RP/0//CPU0:ios(config-12vpn-xf)# neighbor evpn evi 100 target 3 source 4

Configure BFD on the Customer Edge Router
Configure BFD on the customer edge router.

RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# router bgp 100
RP/0//CPU0:router(config-bgp)# bgp router-id 10.10.10.1
RP/0//CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0//CPU0:router(config-bgp-af)# exit
RP/0//CPU0:router(config-bgp)# neighbor 172.16.0.1
RP/0//CPU0:router(config-bgp-nbr)# remote-as 100
RP/0//CPU0:router(config-bgp-nbr)# bfd fast-detect
RP/0//CPU0:router(config-bgp-nbr)# bfd multiplier 2
RP/0//CPU0:router(config-bgp-nbr)# bfd minimum-interval 100
RP/0//CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0//CPU0:router(config-bgp-nbr-af)#

Running Configuration
This section shows the BFD Transparency configuration.

! interface TenGigE 0/4/0/2/0.1
  l2transport
  router isis 1
    is-type level-2-only
    net 49.0000.1000.0000.0001.00
    nsr
    nsf cisco
    address-family ipv4 unicast
    metric-style wide
!
interface Bundle-Ether199
  address-family ipv4 unicast
interface Loopback0
  address-family ipv4 unicast
router bgp 100
  bgp router-id 10.10.10.1
  address-family ipv4 unicast
  neighbor 192.0.2.1
  remote-as 100
update-source Loopback 0
address-family l2vpn evpn
!
mpls ldp
interface Bundle-Ether199
!
l2vpn
router-id 10.10.10.1
xconnect group bfdtr
p2p vpws-ce
interface TenGigE 0/4/0/2/0.1
neighbor evpn evi 100 target 3 source 4

router bgp 100
bgp router-id 10.10.10.1
address-family ipv4 unicast
!
neighbor 172.16.0.1
address-family ipv4 unicast
remote-as 100
bfd fast-detect
bfd multiplier 2
bfd minimum-interval 100
update-source TenGigE0/0/0/16.1
address-family ipv4 unicast

Verification

The show outputs given in the following section display the details of the configuration of the BFD transparency, and the status of their configuration.

/* 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
-------------- ---------- ------------------------ -------- ---- ----- ---
Te0/0/0/4.1 10.1.1.1 0s(0s*0) 20ms(10ms*2) UP Yes 0/0/CPU0

/* Verify if the BFD session is up and check the timer value, numbers of hellos exchanged, and information about last packet. */
RP/0//CPU0:router# show bfd session destination 10.1.1.1 detail
Thu Jan 4 03:09:48.573 UTC
I/f: TenGigE0/0/0/4.1, Location: 0/0/CPU0
Dest: 10.1.1.1
Src: 10.1.1.2
State: UP for 0d:0h:9m:27s, number of times UP: 1
Session type: PR/V4/SH
Received parameters:
  Version: 1, desired tx interval: 10 ms, required rx interval: 10 ms
  Required echo rx interval: 0 ms, multiplier: 2, diag: None
  My discr: 2147483898, your discr: 2147483899, state UP, D/F/P/C/A: 0/0/0/1/0
Transmitted parameters:
  Version: 1, desired tx interval: 10 ms, required rx interval: 10 ms
  Required echo rx interval: 0 ms, multiplier: 2, diag: None
  My discr: 2147483899, your discr: 2147483898, state UP, D/F/P/C/A: 0/1/0/1/0
Timer Values:
Local negotiated async tx interval: 10 ms
Remote negotiated async tx interval: 10 ms
Desired echo tx interval: 0 s, local negotiated echo tx interval: 0 ms
Echo detection time: 0 ms (0 ms * 2), async detection time: 20 ms (10 ms * 2)

Local Stats:
Intervals between async packets:
  Tx: Number of intervals=100, min=6 ms, max=6573 ms, avg=1506 ms
  Last packet transmitted 186 s ago
  Rx: Number of intervals=100, min=4 ms, max=5 s, avg=575 ms
  Last packet received 184 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:

<table>
<thead>
<tr>
<th>Client</th>
<th>Desired Interval</th>
<th>Multiplier</th>
<th>Adjusted Interval</th>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>bgp-default</td>
<td>10 ms</td>
<td>2</td>
<td>10 ms</td>
<td>2</td>
</tr>
</tbody>
</table>

H/W Offload Info:
H/W Offload capability : Y, Hosted NPU : 0//CPU0
Async Offloaded : Y, Echo Offloaded : N
Async rx/tx : 344/209

Platform Info:
NPU ID: 0
Async RTC ID : 1 Echo RTC ID : 0
Async Feature Mask : 0x0 Echo Feature Mask : 0x0
Async Session ID : 0xfb Echo Session ID : 0x0
Async Tx Key : 0x800000fb 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 the complete history including session state, type, transitions, offload history, last down reason if any, received and transmitted packets, rx/tx intervals, location, timestamp, and local and remote descriptors. */

RP/0/RP0/CPU0:router# show bfd session status history destination 10.1.10.1 loc$

Thu Jan 4 03:45:18.768 UTC
I/f: TenGigE0/0/0/4.10, Location: 0//CPU0 table_id:0xe0000000
State: UP, flags:0x80040
Ifotype: 0x19, basecaps: 107
Async dest addr: 10.1.10.1
Async src addr: 10.1.10.2
Echo dest addr: 10.1.10.2
Echo src addr: 192.0.2.1
Additional info from Flags:
  FIB is READY
  Session Active on 0/0/CPU0
Platform Info: 0x0, Mac Length: 18
Redundancy session info:
  Created from active BFD server
Last Down Diag: None
Last UP Time: Jan 4 03:00:19.272

Received parameters:
  Version: 1, desired tx interval: 10 ms, required rx interval: 10 ms
Bidirectional Forwarding Detection over Logical Bundle

Bidirectional Forwarding Detection (BFD) over Logical Bundle (BLB) feature implements and deploys BFD over bundle VLAN interfaces.

BLB feature is different from the BFD over Bundle (BoB) feature which runs on non-VLAN bundle interfaces. These two features are distinct from each other. Do not configure these two features at the same time on the same bundle.

Routing protocols are BFD clients for the BLB feature. Hence if BFD session goes down, it will bring down the routing session.
Restriction

BLB sessions are restricted to an interval of 300 milliseconds and a multiplier of 3. Though you can configure more aggressive parameters, Cisco does not recommend it.

Configuration

Perform the following tasks to configure the BLB feature.

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

Create VLAN sub-interface under bundle interface

```conf
Router# configure
Router(config)# interface Bundle-Ether 2.1
Router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
Router(config-if)# encapsulation dot1q 1
Router(config-if)# end
```

Enable BFD on a static route.

```conf
Router# configure
Router(config)# router static
Router(config-static)# address-family ipv4 unicast
Router(config-static)# 10.158.3.13/32 10.1.1.2 bfd fast-detect minimum-interval 300 multiplier 3
```

Enable BFD on IS-IS.

```conf
Router# configure
Router(config)# router isis cybi
Router(config-isis)# interface Bundle-Ether 2.1
Router(config-isis-if)# bfd minimum-interval 300
Router(config-isis-if)# bfd multiplier 3
Router(config-isis-if)# bfd fast-detect ipv4
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# end
```

Enable BFD for OSPF on an interface.

```conf
Router# configure
Router(config)# router ospf cybi
Router(config-ospf)# area 0
Router(config-ospf)# interface Bundle-Ether 2.1
Router(config-ospf-if)# bfd fast-detect
Router(config-ospf-if)# bfd minimum-interval 300
Router(config-ospf-if)# bfd multiplier 3
```
Enable BFD on a BGP neighbor

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

Configure a specific LC (or LCs) to host BLB sessions. The BLB sessions and bundle member links need not be configured on the same LC. For example, you can configure the bundle member links on LC slot 2 and slot 3 while you configure BLB sessions to be hosted on LC slot 5.

```
Router(config)# bfd
Router(config-bfd)# multipath include location 0/6/CPU0
Router(config-bfd)# multipath include location 0/2/CPU0
```

Running Configuration

This section shows the BFD over logical bundle configuration.

```
interface Bundle-Ether2.1
  ipv4 address 10.1.1.1 255.255.255.0
  encapsulation dot1q 1
!
router static
  address-family ipv4 unicast
    10.158.3.13/32 10.1.1.2 bfd fast-detect minimum-interval 300 multiplier 3
  !
!
router isis cybi
  interface Bundle-Ether2.1
    bfd minimum-interval 300
    bfd multiplier 3
    bfd fast-detect ipv4
    address-family ipv4 unicast
    !
  !
router ospf cybi
  area 0
    interface Bundle-Ether2.1
      bfd fast-detect
      bfd minimum-interval 300
      bfd multiplier 3
    !
  !
bfdf
  multipath include location 0/6/CPU0
  multipath include location 0/2/CPU0
```
Verification

The show outputs given in the following section display the details of the configuration of the BLB feature, and the status of their configuration.

/* Verify the BFD session summary information. */

RP/0/RSP0/CPU0:router# show bfd session

<table>
<thead>
<tr>
<th>Interface</th>
<th>Dest Addr</th>
<th>Local det time(int*mult)</th>
<th>State</th>
<th>Echo</th>
<th>Async</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE2.1</td>
<td>10.158.1.2</td>
<td>0s(0s<em>0) 900ms(300ms</em>3)</td>
<td>UP</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>0/6/CPU0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BE2.2</td>
<td>10.158.2.2</td>
<td>0s(0s<em>0) 900ms(300ms</em>3)</td>
<td>UP</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>0/6/CPU0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/* Verify the BFD session detail information for the specified interface. */

RP/0/RSP0/CPU0:router# show bfd session detail interface Bundle-Ether 2.1

I/f: Bundle-Ether2.1, Location: 0/6/CPU0
Dest: 10.158.1.2
Src: 10.158.1.1
State: UP for 0d:21h:35m:54s, number of times UP: 1
Session type: SW/V4/SH/BL
Received parameters:
Version: 1, desired tx interval: 300 ms, required rx interval: 300 ms
Required echo rx interval: 0 ms, multiplier: 3, diag: None
My discr: 12584150, your discr: 845, state UP, D/F/P/C/A: 0/0/0/1/0
Transmitted parameters:
Version: 1, desired tx interval: 300 ms, required rx interval: 300 ms
Required echo rx interval: 0 ms, multiplier: 3, diag: None
My discr: 845, your discr: 12584150, state UP, D/F/P/C/A: 0/1/0/1/0
Timer Values:
Local negotiated async tx interval: 300 ms
Remote negotiated async tx interval: 300 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: 900 ms(300 ms*3)
Label:
Internal label: 64119/0xfa77
Local Stats:
Intervals between async packets:
Tx: Number of intervals=3, min=160 ms, max=726 ms, avg=385 ms
Last packet transmitted 77754 s ago
Rx: Number of intervals=4, min=100 ms, max=270 ms, avg=183 ms
Last packet received 77753 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: Dec 14 18:38:06.721

Session owner information:

<table>
<thead>
<tr>
<th>Desired</th>
<th>Adjusted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Interval</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>ospf-cybi</td>
<td>300 ms</td>
</tr>
<tr>
<td>ipv4_static</td>
<td>300 ms</td>
</tr>
</tbody>
</table>

H/W Offload Info:
H/W Offload capability : Y, Hosted NPU : 0/6/CPU0
Async Offloaded : Y, Echo Offloaded : N
Async rx/tx : 5/4

Platform Info:
NPU ID: 0
Async RTC ID : 1
Echo RTC ID : 0
Async Feature Mask : 0x0
Echo Feature Mask : 0x0
Async Session ID : 0x34d
Echo Session ID : 0x0
Async Tx Key : 0x34d
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 the BFD session detail information for the specified IP address. */

RP/0/RSP0/CPU0# show bfd session detail destination 10.158.2.2

I/f: Bundle-Ether2.2, Location: 0/6/CPU0
Dest: 10.158.1.2
Src: 10.158.1.1
State: UP for 0d:21h:39m:36s, number of times UP: 1
Session type: SW/V4/SH/BL
Received parameters:
Version: 1, desired tx interval: 300 ms, required rx interval: 300 ms
Required echo rx interval: 0 ms, multiplier: 3, diag: None
My discr: 12584129, your discr: 824, state UP, D/F/P/C/A: 0/0/0/1/0
Transmitted parameters:
Version: 1, desired tx interval: 300 ms, required rx interval: 300 ms
Required echo rx interval: 0 ms, multiplier: 3, diag: None
My discr: 824, your discr: 12584129, state UP, D/F/P/C/A: 0/1/0/1/0
Timer Values:
Local negotiated async tx interval: 300 ms
Remote negotiated async tx interval: 300 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: 900 ms(300 ms*3)
Label:
Internal label: 64098/0xfa62
Local Stats:
Intervals between async packets:
Tx: Number of intervals=3, min=160 ms, max=616 ms, avg=383 ms
Last packet transmitted 77975 s ago

Routing Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
Rx: Number of intervals=4, min=100 ms, max=374 ms, avg=209 ms
Last packet received 77975 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: Dec 14 18:38:06.721
Session owner information:
<table>
<thead>
<tr>
<th>Client</th>
<th>Desired Interval</th>
<th>Multiplier</th>
<th>Adjusted Interval</th>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>isis-cybi</td>
<td>300 ms</td>
<td>3</td>
<td>300 ms</td>
<td>3</td>
</tr>
<tr>
<td>bgp-default</td>
<td>300 ms</td>
<td>3</td>
<td>300 ms</td>
<td>3</td>
</tr>
</tbody>
</table>
H/W Offload Info:
H/W Offload capability : Y, Hosted NPU : 0/6/CPU0
Async Offloaded : Y, Echo Offloaded : N
Async rx/tx : 5/4
Platform Info:
NPU ID: 0
Async RTC ID : 1  Echo RTC ID : 0
Async Feature Mask : 0x0  Echo Feature Mask : 0x0
Async Session ID : 0x338  Echo Session ID : 0x0
Async Tx Key : 0x338  Echo Tx Key : 0x0
async Tx Stats addr : 0x0  Echo Tx Stats addr : 0x0
Async Rx Stats addr : 0x0  Echo Rx Stats addr : 0x0

IPv4 Multihop BFD

IPv4 Multihop BFD is a BFD session between two addresses between two nodes. 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 network hops.

The IPv4 Multihop BFD feature provides sub-second forwarding failure detection for a destination more than one hop, and up to 255 hops, away. The `bfd multihop ttl-drop-threshold` command can be used to drop BFD packets coming from neighbors exceeding a certain number of hops. IPv4 Multihop BFD feature is supported on all currently supported media-type for BFD singlehop.

A BFD multihop session is set up between a unique source-destination address pair provided by the client. A session can be set up between two endpoints that have IP connectivity.

IPv4 Multihop BFD feature runs only on a default VRF and the multihop destination must resolve in a pure IP path. NCS 5500 router doesn't support the multihop destination that resolves in labeled or MPLS path.

Configure IPv4 Multihop BFD

This section describes how you can configure IPv4 Multihop BFD feature.

```
Router# configure
Router(config)# bfd
Router(config)# multihop ttl-drop-threshold 225
```
Verifier 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)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# commit
```

**Running Configuration**

```
bfd
  multipath ttl-drop-threshold 225
  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
    route-policy PASS-ALL in
    route-policy PASS-ALL out
!`
```

**Verification**

The show outputs given in the following section display the details of the configuration of the IPv4 Multihop BFD feature, and the status of their configuration.

```
Router# show tech-support bfdhwoff location 0/7/CPU0 file
  harddisk:
  Tue Mar 20 11:20:29.214 PDT
++ Show tech start time: 2018-Mar-20.112029.PDT ++
  Tue Mar 20 11:20:30 PDT 2018 Waiting for gathering to complete .........................
  Tue Mar 20 11:22:37 PDT 2018 Compressing show tech output Show tech output available at
  0/RP0/CPU0 :
++ Show tech end time: 2018-Mar-20.112237.FDT ++
```
Implementing UCMP

The unequal cost multipath (UCMP) load-balancing provides the capability to load balance traffic proportionally across multiple paths, with different cost. Generally, higher bandwidth paths have lower Interior Gateway Protocol (IGP) metrics configured, so that they form the shortest IGP paths.

With the UCMP load-balancing enabled, protocols can use even lower bandwidth paths or higher cost paths for traffic, and can install these paths to the forwarding information base (FIB). These protocols still install multiple paths to the same destination in FIB, but each path will have a 'load metric/weight' associated with it. FIB uses this load metric/weight to decide the amount of traffic that needs to be sent on a higher bandwidth path and the amount of traffic that needs to be sent on a lower bandwidth path.

In the following example, there are 3 paths to get to Network X as follows:

*Figure 7: Topology for UCMP*

<table>
<thead>
<tr>
<th>Paths</th>
<th>Cost from Router E to Net -X</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-B-A</td>
<td>40</td>
</tr>
<tr>
<td>E-C-A</td>
<td>50</td>
</tr>
<tr>
<td>E-D-A</td>
<td>40</td>
</tr>
</tbody>
</table>
IGP selects the lowest path links, i.e E-B-A and E-D-A. The path E-C-A is not considered for load balancing because of higher cost. The lowest path link E-D (5) is not a tie breaker, as the end to end cost to the Network X is considered.

- ECMP vs. UCMP Load Balancing, on page 230
- UCMP Minimum Integer Ratio, on page 230
- Configuring IS-IS With Weight, on page 231
- Configuring IS-IS With Metric, on page 232
- Configuring BGP With Weights, on page 233
- Configuring TE Tunnel With Weights, on page 234
- Policy-Based Tunnel Selection, on page 235

**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**

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 0x8, source rib (7), 0 backups
[14 type 3 flags 0x8401 (0x71b02d90) ext 0x0 (0x0)]
LWI-LDI[type=3, refc=1, ptr=0x71b98870, sh-l dia]=0x71b98870
gateway array update type-time 1 Oct 16 06:34:46.196
LDI Update time Oct 16 06:34:46.197
```
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-router)# address-family ipv4 unicast
RP/0/RSP0/CPU0# router (config-router-af)# metric 1
```

**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, \(21 = (1 \text{ configured} + 20 \text{ added by IS-IS})\), weight \(0xFFFFFFF (4294967295)\)

The UCMP path route metric, \(120 = (100 + 20)\), weight \(= (21/120) \times 4294967295 = 751619276\)

GCD is one. So Normalized Weight is:

\[
(4294967295 \times 64) / (4294967295 + 751619276) = 54
\]

\[
(751619276 \times 64) / (4294967295 + 751619276) = 9
\]

**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
```
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 in
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 (?), 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 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] via 1.0.0.2/32, recursive
[1] via 2.0.0.2/32, recursive
```

**Configuring TE Tunnel With Weights**

Use the `load-share` command on `tunnel-te` config to set weight.

```
RP/0/RSP0/CPU0:router(config)# interface tunnel-te1
RP/0/RSP0/CPU0:router(config-if)# load-share 8
```

**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, which is 64 for ASR9K. UCMP with PBTS can have more total_paths (buckets) than the supported number of paths (buckets) for all Forwarding Classes (FCs), which is 64 for ASR9K.

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)]
LDI Update time Mar 7 05:46:11.118
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
Routing Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x

Implementing UCMP

Policy-Based Tunnel Selection

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 [0x997b4f998 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 [0x997b4fee8 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 [0x997b505d0 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 [0x997b4fc40 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 [0x997b50988 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 [0x997b50080 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 [0x97b4fd50 0x0]
next hop VRF - 'default', table - 0xe0000000
next hop ::ffff:200.0.0.1/128
local adjacency
labels imposed {ExpNullv6}
Routing Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x

Implementing UCMP

Policy-Based Tunnel Selection

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]
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 [0x97b51ff50 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 4
slot 24, weight 1, normalized_weight 1, forward class 4
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
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)

<table>
<thead>
<tr>
<th>Hash</th>
<th>OK</th>
<th>Interface</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Y</td>
<td>tunnel-te57</td>
<td>point2point</td>
</tr>
<tr>
<td>1</td>
<td>Y</td>
<td>tunnel-te58</td>
<td>point2point</td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
<td>tunnel-te59</td>
<td>point2point</td>
</tr>
<tr>
<td>3</td>
<td>Y</td>
<td>tunnel-te60</td>
<td>point2point</td>
</tr>
<tr>
<td>4</td>
<td>Y</td>
<td>tunnel-te61</td>
<td>point2point</td>
</tr>
<tr>
<td>5</td>
<td>Y</td>
<td>tunnel-te62</td>
<td>point2point</td>
</tr>
<tr>
<td>6</td>
<td>Y</td>
<td>tunnel-te63</td>
<td>point2point</td>
</tr>
<tr>
<td>7</td>
<td>Y</td>
<td>tunnel-te8</td>
<td>point2point</td>
</tr>
<tr>
<td>8</td>
<td>Y</td>
<td>tunnel-te1</td>
<td>point2point</td>
</tr>
<tr>
<td>9</td>
<td>Y</td>
<td>tunnel-te2</td>
<td>point2point</td>
</tr>
<tr>
<td>10</td>
<td>Y</td>
<td>tunnel-te3</td>
<td>point2point</td>
</tr>
<tr>
<td>11</td>
<td>Y</td>
<td>tunnel-te4</td>
<td>point2point</td>
</tr>
<tr>
<td>12</td>
<td>Y</td>
<td>tunnel-te5</td>
<td>point2point</td>
</tr>
<tr>
<td>13</td>
<td>Y</td>
<td>tunnel-te6</td>
<td>point2point</td>
</tr>
<tr>
<td>14</td>
<td>Y</td>
<td>tunnel-te7</td>
<td>point2point</td>
</tr>
<tr>
<td>15</td>
<td>Y</td>
<td>tunnel-te16</td>
<td>point2point</td>
</tr>
<tr>
<td>16</td>
<td>Y</td>
<td>tunnel-te9</td>
<td>point2point</td>
</tr>
<tr>
<td>17</td>
<td>Y</td>
<td>tunnel-te10</td>
<td>point2point</td>
</tr>
<tr>
<td>18</td>
<td>Y</td>
<td>tunnel-te11</td>
<td>point2point</td>
</tr>
<tr>
<td>19</td>
<td>Y</td>
<td>tunnel-te12</td>
<td>point2point</td>
</tr>
<tr>
<td>20</td>
<td>Y</td>
<td>tunnel-te13</td>
<td>point2point</td>
</tr>
<tr>
<td>21</td>
<td>Y</td>
<td>tunnel-te14</td>
<td>point2point</td>
</tr>
<tr>
<td>22</td>
<td>Y</td>
<td>tunnel-te15</td>
<td>point2point</td>
</tr>
<tr>
<td>23</td>
<td>Y</td>
<td>tunnel-te24</td>
<td>point2point</td>
</tr>
<tr>
<td>24</td>
<td>Y</td>
<td>tunnel-te17</td>
<td>point2point</td>
</tr>
<tr>
<td>25</td>
<td>Y</td>
<td>tunnel-te18</td>
<td>point2point</td>
</tr>
<tr>
<td>26</td>
<td>Y</td>
<td>tunnel-te19</td>
<td>point2point</td>
</tr>
<tr>
<td>27</td>
<td>Y</td>
<td>tunnel-te20</td>
<td>point2point</td>
</tr>
<tr>
<td>28</td>
<td>Y</td>
<td>tunnel-te21</td>
<td>point2point</td>
</tr>
<tr>
<td>29</td>
<td>Y</td>
<td>tunnel-te22</td>
<td>point2point</td>
</tr>
<tr>
<td>30</td>
<td>Y</td>
<td>tunnel-te23</td>
<td>point2point</td>
</tr>
<tr>
<td>31</td>
<td>Y</td>
<td>tunnel-te32</td>
<td>point2point</td>
</tr>
<tr>
<td>32</td>
<td>Y</td>
<td>tunnel-te25</td>
<td>point2point</td>
</tr>
<tr>
<td>33</td>
<td>Y</td>
<td>tunnel-te26</td>
<td>point2point</td>
</tr>
<tr>
<td>34</td>
<td>Y</td>
<td>tunnel-te27</td>
<td>point2point</td>
</tr>
<tr>
<td>35</td>
<td>Y</td>
<td>tunnel-te28</td>
<td>point2point</td>
</tr>
<tr>
<td>36</td>
<td>Y</td>
<td>tunnel-te29</td>
<td>point2point</td>
</tr>
<tr>
<td>37</td>
<td>Y</td>
<td>tunnel-te30</td>
<td>point2point</td>
</tr>
<tr>
<td>38</td>
<td>Y</td>
<td>tunnel-te31</td>
<td>point2point</td>
</tr>
<tr>
<td>39</td>
<td>Y</td>
<td>tunnel-te40</td>
<td>point2point</td>
</tr>
<tr>
<td>40</td>
<td>Y</td>
<td>tunnel-te33</td>
<td>point2point</td>
</tr>
<tr>
<td>41</td>
<td>Y</td>
<td>tunnel-te34</td>
<td>point2point</td>
</tr>
<tr>
<td>42</td>
<td>Y</td>
<td>tunnel-te35</td>
<td>point2point</td>
</tr>
<tr>
<td>43</td>
<td>Y</td>
<td>tunnel-te36</td>
<td>point2point</td>
</tr>
<tr>
<td>44</td>
<td>Y</td>
<td>tunnel-te37</td>
<td>point2point</td>
</tr>
<tr>
<td>45</td>
<td>Y</td>
<td>tunnel-te38</td>
<td>point2point</td>
</tr>
<tr>
<td>46</td>
<td>Y</td>
<td>tunnel-te39</td>
<td>point2point</td>
</tr>
<tr>
<td>47</td>
<td>Y</td>
<td>tunnel-te44</td>
<td>point2point</td>
</tr>
<tr>
<td>48</td>
<td>Y</td>
<td>tunnel-te45</td>
<td>point2point</td>
</tr>
<tr>
<td>49</td>
<td>Y</td>
<td>tunnel-te46</td>
<td>point2point</td>
</tr>
<tr>
<td>50</td>
<td>Y</td>
<td>tunnel-te47</td>
<td>point2point</td>
</tr>
<tr>
<td>51</td>
<td>Y</td>
<td>tunnel-te48</td>
<td>point2point</td>
</tr>
<tr>
<td>52</td>
<td>Y</td>
<td>tunnel-te41</td>
<td>point2point</td>
</tr>
<tr>
<td>53</td>
<td>Y</td>
<td>tunnel-te42</td>
<td>point2point</td>
</tr>
<tr>
<td>54</td>
<td>Y</td>
<td>tunnel-te43</td>
<td>point2point</td>
</tr>
<tr>
<td>55</td>
<td>Y</td>
<td>tunnel-te56</td>
<td>point2point</td>
</tr>
<tr>
<td>56</td>
<td>Y</td>
<td>tunnel-te49</td>
<td>point2point</td>
</tr>
<tr>
<td>57</td>
<td>Y</td>
<td>tunnel-te50</td>
<td>point2point</td>
</tr>
<tr>
<td>58</td>
<td>Y</td>
<td>tunnel-te51</td>
<td>point2point</td>
</tr>
<tr>
<td>59</td>
<td>Y</td>
<td>tunnel-te52</td>
<td>point2point</td>
</tr>
<tr>
<td>60</td>
<td>Y</td>
<td>tunnel-te53</td>
<td>point2point</td>
</tr>
<tr>
<td>61</td>
<td>Y</td>
<td>tunnel-te54</td>
<td>point2point</td>
</tr>
<tr>
<td>62</td>
<td>Y</td>
<td>tunnel-te55</td>
<td>point2point</td>
</tr>
</tbody>
</table>
Chapter 9

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 249
- Restrictions for Fast Reroute with Loop-Free Alternate, on page 249
- IS-IS and FRR, on page 250
- Repair Paths, on page 250
- LFA Overview, on page 250
- LFA Calculation, on page 251
- Interaction Between RIB and Routing Protocols, on page 251
- Fast Reroute with Remote Loop-Free Alternate, on page 252
- Configuration , on page 253

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**

```bash
/* 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 0/0/0/0
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if)# fast-reroute per-prefix
Router(config-isis-if)# exit

/* 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 HundredGigE0/2/0/0
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.**

```bash
/* 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
```
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 HundredGigE0/2/0/0
  point-to-point
  address-family ipv4 unicast
  fast-reroute per-prefix

/* FRR with local LFA with OSPF */
routing ospf 50
  router-id 10.1.1.1
  address-family ipv4 unicast
  area 0

Routing Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.3.x
mpls traffic-eng
interface Loopback0
!
interface HundredGigE0/2/0/0
fast-reroute per-prefix
!
!
This section shows the FRR with remote LFA configuration.

/* FRR with remote LFA with ISIS */
* 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 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 HundredGigE0/2/0/0
fast-reroute per-prefix
fast-reroute per-prefix remote-lfa tunnel mpls-ldp
!
!
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//CP00: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 protocols.

/* Verify the MPLS LDP configuration. */

RP/0/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

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Label</th>
<th>Label(s)</th>
<th>Outgoing</th>
<th>Next Hop</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In</td>
<td>Out</td>
<td>Interface</td>
<td></td>
<td>G S R</td>
</tr>
<tr>
<td>192.0.2.0/24</td>
<td>16019</td>
<td>{ 16001</td>
<td>Te0/0/0/0/0</td>
<td>10.1.1.1</td>
<td>(!)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28006</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>192.0.2.1/32</td>
<td>16013</td>
<td>ImpNull</td>
<td>Te0/7/0/3</td>
<td>192.0.2.1</td>
<td></td>
</tr>
<tr>
<td>192.0.1.0/32</td>
<td>16014</td>
<td>{ 16001</td>
<td>Te0/0/0/0/0</td>
<td>10.1.1.1</td>
<td>(!)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16002</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.9.9.9/32</td>
<td>16012</td>
<td>16001</td>
<td>Te0/0/0/0/0</td>
<td>10.1.1.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>28006</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.23.1.0/24</td>
<td>16018</td>
<td>16004</td>
<td>Te0/7/0/3</td>
<td>192.0.2.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ImpNull</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.34.1.0/24</td>
<td>16015</td>
<td>ImpNull</td>
<td>Te0/0/0/0/0</td>
<td>10.1.1.1</td>
<td></td>
</tr>
<tr>
<td>10.0.0.1/32</td>
<td>16011</td>
<td>{ 16001</td>
<td>Te0/0/0/0/0</td>
<td>10.1.1.1</td>
<td>(!)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16013</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.100.0.2/32</td>
<td>16010</td>
<td>{ 16001</td>
<td>Te0/7/0/3</td>
<td>192.0.2.1</td>
<td></td>
</tr>
</tbody>
</table>

Implementing Fast Reroute Loop-Free Alternate Verification