Cisco NCS 560 シリーズ ルータ（IOS XR リリース 7.0.x）ルーティング コンフィギュレーション ガイド

初版：2019 年 8 月 30 日

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

目 次

第 1 章

IS-IS の実装 1

IS-IS のイネーブル化およびレベル 1 またはレベル 2 ルーティングの設定 1

単一トポロジ IPv6 3

IS-IS のシングル トポロジの設定 3

単一トポロジ構成用の SPF 間隔の設定 8

IS-IS のルートのカスタマイズ 10

RIB にプレフィックスを追加するためのプライオリティの設定 13

IS-IS のインターフェイス 14

IS-IS インターフェイス ルートのタギング 15

LSP フラッディングの制限 17

IS-IS の LSP フラッディングの制御 17

残りの最小ライフタイム 21

IS-IS 認証 22

IS-IS の認証の設定 22

IS-IS のキーチェーンの設定 24

ノンストップ フォワーディング 25

IS-IS のノンストップ フォワーディングの設定 26

ISIS NSR 28

ISIS-NSR の設定 28

マルチプロトコル ラベル スイッチング トラフィック エンジニアリング 30

IS-IS の MPLS トラフィック エンジニアリングの設定 30

MPLS TE 転送隣接 32

IS-IS の隣接関係の調整 32

MPLS ラベル配布プロトコル IGP 同期 35

MPLS LDP IS-IS 同期の設定 35
IS-IS 過負荷ビット無効化 36
IS-IS 過負荷ビット無効化の設定 37
IS-IS の参照 38
IS-IS 機能の概要 38
デフォルト ルート 39
ルータの過負荷ビット 39
マルチトポロジ動作中の過負荷ビット設定 39
IS-IS インスタンスの attached ビット 40
ルート タグの IS-IS サポート 40
特定のインターフェイスでのフラッディングのブロック 40
最大 LSP ライフタイムおよび更新間隔 40
メッシュ グループの設定 41
マルチインスタンス IS-IS 41
ラベル配布プロトコル IGP 自動設定 41
LDP グレースフル リスタートとの MPLS LDP-IGP 同期の互換性 42
IGP ノンストップ フォワーディングとの MPLS LDP-IGP 同期の互換性 42

第 2 章 **OSPF の実装** 43
OSPF の実装の前提条件 44
OSPF のイネーブル化 44
OSPF の設定と動作の確認 47
スタブ エリア 49
Not-So-Stubby Area 49
スタブ エリア タイプおよび Not-So-Stubby Area タイプの設定 50
OSPF のネイバーおよび隣接関係 52
非ブロードキャスト ネットワークのネイバーの設定 52
認証方法 56
OSPF Version 2 の異なる階層レベルでの認証の設定 56
OSPF に同じ LSA が生成される頻度または受け入れられる頻度の制御 60
OSPF の仮想リンクおよび中継エリア 61

仮想リンクの作成 62
OSPF ABR でのサブネットワーク LSA の集約 67
OSPF のルート再配布 69
OSPF へのルートの再配布 69
OSPF Version 2 のノンストップ フォワーディング 72
Cisco for OSPF Version 2 固有のノンストップ フォワーディングの設定 72
OSPF Shortest Path First スロットリング 74
OSPF Shortest Path First スロットリングの設定 75
OSPFv3 のグレースフル リスタート 77
OSPFv3 グレースフル リスタートの設定 78
グレースフル リスタートに関する情報の表示 79
OSPF Version 2 のウォーム スタンバイとノンストップ ルーティング 80
OSPFv2 のノンストップ ルーティングのイネーブル化 80
OSPF Version 3 のウォーム スタンバイとノンストップ ルーティング 81
OSPFv3 のノンストップ ルーティングのイネーブル化 81
OSPFv2OSPF SPF プレフィックスのプライオリティ設定 82
OSPFv2 OSPF SPF プレフィックス プライオリティの設定 83
プロバイダー エッジからカスタマー エッジ（PE-CE）プロトコルとしての OSPF の設定 86
複数の OSPF インスタンスの作成（OSPF プロセスおよび VRF） 88
OSPF のラベル配布プロトコル IGP 自動設定 90
OSPF のラベル配布プロトコル IGP 自動設定の設定 90
LDP IGP 同期の設定：OSPF 91
OSPF 認証のメッセージ ダイジェスト管理 93
OSPF の認証メッセージ ダイジェスト管理の設定 93
OSPF の GTSM TTL セキュリティ メカニズム 96
OSPF の一般 TTL セキュリティ メカニズム（GTSM）の設定 97
OSPF の参照 99
OSPF 機能の概要 99
Cisco IOS XR ソフトウェアの OSPFv3 と OSPFv2 の比較 101
OSPF の階層 CLI および CLI 継承 101

OSPF ルーティング コンポーネント 102
自律システム 102
エリア 102
ルータ 103
OSPF プロセスおよびルータ ID 104
サポート対象 OSPF ネットワーク タイプ 104
OSPF のルート認証方法 105
プレーン テキスト認証 105
MD5 認証 105
キー ロールオーバー 105
OSPF FIB ダウンロード通知 106
OSPF の指定ルータ（DR） 106
OSPF のデフォルト ルート 106
OSPF Version 2 のリンクステート アドバタイズメント タイプ 106
OSPFv3 のリンクステート アドバタイズメント タイプ 108
パッシブ インターフェイス 109
グレースフル リスタート操作のモード 109
リスタート モード 110
ヘルパー モード 110
プロトコル シャットダウン モード 111
OSPF Version 2 および OSPFv3 でのロード バランシング 112
OSPFv2 のパス計算要素 112
OSPFv3 の管理情報ベース（MIB） 113
OSPFv2 の VRF-lite サポート 113
OSPFv3 タイマーの更新 114

第 3 章 **RIB の実装とモニタリング** 115
ルーティング テーブルを使用した RIB 設定の確認 116
ネットワーキングとルーティングの問題の検証 117
RIB ネクストホップ ダンプニングのディセーブル化 118
RCC および LCC オンデマンド スキャンのイネーブル化 119

RCC および LCC バックグラウンド スキャンのイネーブル化 120
RIB の参照 122
BGP およびその他のプロトコルでの RIB データ構造 122
RIB アドミニストレーティブ ディスタンス 122
RIB 統計情報 123
RIB 隔離 124
ルートとラベルの整合性チェッカ 124

第 4 章 ルーティング ポリシーの実装 127
ルーティング ポリシー実装の制約事項 127
ルート ポリシーの定義 128
BGP ネイバーへのルーティング ポリシーのアタッチ 130
テキスト エディタを使用したルーティング ポリシーの変更 131
ルーティング ポリシーの参照 134
ルーティング ポリシー言語 134
ルーティング ポリシー言語の概要 134
ルーティング ポリシー言語の構造 135
ルーティング ポリシー言語コンポーネント 143
ルーティング ポリシー言語使用方法 143
ポリシー定義 146
パラメータ化 147
接続点でのパラメータ化 147
グローバル パラメータ化 148
ポリシー適用のセマンティック 149
ブール演算子優先 149
同じ属性の複数の変更 149
属性を変更するとき 150
デフォルトのドロップ処理 151
制御フロー 151
ポリシー検証 152
ポリシー ステートメント 154

注記 154
処理 155
アクション 157
If 157
ブール条件 158
apply 160
接続点 160
BGP ポリシー接続点 161
OSPF ポリシー接続点 181
OSPFv3 ポリシー接続点 185
IS-IS ポリシー接続点 186
ルーティング ポリシーの非破壊編集 188
アタッチされたポリシーの変更 188
アタッチされないポリシーの変更 188
ルーティング ポリシー設定要素の編集 189
階層型ポリシー条件 191
条件ポリシーの適用 191
ネストされたワイルドカード適用ポリシー 194
VRF インポート ポリシーの強化 195
集約されたルートの照合 195
着信ポリシーでのプライベート AS の削除 196

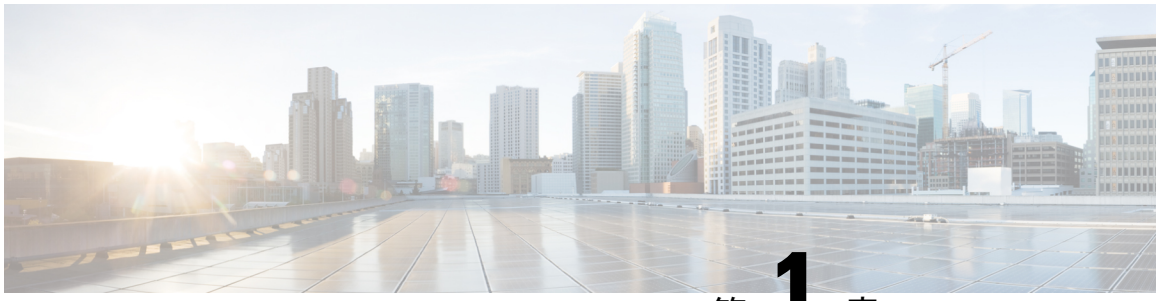
第 5 章 スタティック ルートの実装 197
スタティック ルートの実装に関する制約事項 197
スタティック ルートの設定 198
フローティング スタティック ルート 199
フローティング スタティック ルートの設定 199
PE-CE ルータ間でのスタティック ルートの設定 200
IPv4 マルチキャスト スタティック ルート 202
マルチキャスト スタティック ルートの設定 203
デフォルト VRF 204

スタティック ルートを使用した VRF の関連付け 204
スタティック ルートの参照 205
スタティック ルート機能の概要 205
デフォルトのアドミニストレーティブ ディスタンス 206
直接接続されたルート 206
フローティング スタティック ルート 207
完全指定のスタティック ルート 207
再帰スタティック ルート 207

第 6 章 BFD の実装 209
BFD の概要 209
ルータでの BFD の IPv6 チェックサム計算のイネーブル化およびディセーブル化 210
ダイナミック ルーティング プロトコル下での BFD の設定またはスタティック ルートの使用 210
インターフェイスでの OSPF への BFD の有効化 210
インターフェイスでの OSPF3 への BFD の有効化 211
BFD over BGP の有効化 212
IPv4 スタティック ルートでの BFD のイネーブル化 213
IPv6 スタティック ルートでの BFD のイネーブル化 213
BFD カウンタのクリアと表示 214
バンドル上の BFD 214
BFD over Bundle の設定 214
BFD の透過性 216
イーサネット VPN 仮想プライベート ワイヤ サービス 217
設定 217
実行コンフィギュレーション 218
確認 219

第 7 章 Fast Reroute ループフリー代替の実装 223
ループフリー代替による Fast Reroute の前提条件 223
ループフリー代替による Fast Reroute の制約事項 223

IS-IS および FRR 224
修復パス 224
LFA の概要 225
LFA の計算 225
RIB とルーティング プロトコル間の連携 226
リモート ループフリー代替による Fast Reroute 226
設定 228
実行コンフィギュレーション 229
確認 231



第 1 章

IS-IS の実装

Integrated Intermediate System-to-Intermediate System（IS-IS）、インターネット プロトコル バージョン 4（IPv4）は、標準ベースの内部ゲートウェイ プロトコル（IGP）です。Cisco ソフトウェアは、国際標準化機構（ISO）/International Engineering Consortium（IEC）10589 および RFC 1195 に記載されている IP ルーティング機能を実装し、IP バージョン 6（IPv6）向けに標準拡張のシングル トポロジおよびマルチトポロジ IS-IS を追加しています。

このモジュールでは、Cisco IOS XR ネットワークで IS-IS（IPv4 および IPv6）を実装する方法について説明します。

- IS-IS のイネーブル化およびレベル 1 またはレベル 2 ルーティングの設定 （1 ページ）
- 単一トポロジ IPv6 （3 ページ）
- IS-IS のルートのカスタマイズ （10 ページ）
- RIB にプレフィックスを追加するためのプライオリティの設定 （13 ページ）
- IS-IS のインターフェイス （14 ページ）
- LSP フラッディングの制限 （17 ページ）
- IS-IS 認証 （22 ページ）
- ノンストップ フォワーディング （25 ページ）
- ISIS NSR （28 ページ）
- マルチプロトコル ラベル スイッチング トラフィック エンジニアリング （30 ページ）
- IS-IS 過負荷ビット無効化 （36 ページ）
- IS-IS の参照 （38 ページ）

IS-IS のイネーブル化およびレベル 1 またはレベル 2 ルーティングの設定

ここでは、IS-IS をイネーブルにし、エリアのルーティング レベルを設定する方法について説明します。

（注） ステップ 4 のルーティング レベルの設定は任意ですが、適切なレベルの隣接関係を確立するために設定することを強く推奨します。

始める前に

IP アドレスを設定する前に IS-IS を設定できますが、少なくとも 1 つの IP アドレスを設定するまでは IS-IS ルーティングは行われません。

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング インスタンスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

- デフォルトでは、すべての IS-IS インスタンスが自動的にレベル 1 とレベル 2 になります。**is-type** ルータ コンフィギュレーション コマンドを使用して、特定のルーティング インスタンスによって実行されるルーティングのレベルを変更できます。

ステップ 3 **net** *network-entity-title*

例：

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

ルーティング インスタンスの Network Entity Title（NET）を設定します。

- マルチインスタンスの IS-IS を設定する場合は、ルーティング インスタンスごとに NET を指定します。
- この例では、エリア ID が 47.0004.004d.0001 でシステム ID が 0001.0c11.1110.00 のルータを設定します。
- 複数のエリア アドレスを指定するには、追加の NET を指定します。NET のエリア アドレス部分が異なる場合でも、システム ID 部分はすべての設定項目で完全に一致する必要があります。

ステップ 4 **is-type** { **level-1** | **level-1-2** | **level-2-only** }

例：

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

（任意）システム タイプ（エリアまたはバックボーン ルータ）を設定します。

- デフォルトでは、すべての IS-IS インスタンスは **level-1-2** ルータとして動作します。
- **level-1** キーワードは、レベル 1（エリア内）ルーティングのみを実行するようにソフトウェアを設定します。レベル 1 の隣接関係のみが確立されます。ソフトウェアはエリア内の宛先についてのみ学習します。エリア外の宛先を含むすべてのパケットは、エリア内の最も近い **level-1-2** ルータに送信されます。
- **level-2-only** キーワードは、レベル 2（バックボーン）ルーティングのみを実行するようにソフトウェアを設定します。ルータは、他のレベル 2 のみのルータまたは **レベル 1 と 2** のルータとの間でレベル 2 の隣接関係のみを確立します。
- **level-1-2** キーワードは、レベル 1 とレベル 2 の両方のルーティングを実行するようにソフトウェアを設定します。レベル 1 とレベル 2 の両方の隣接関係が確立されます。ルータはレベル 2 バックボーンとレベル 1 エリアの間の境界ルータとして動作します。

ステップ 5 **commit**

ステップ 6 **show isis** [**instance** *instance-id*] **protocol**

例：

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

（任意）IS-IS インスタンスに関するサマリー情報を表示します。

単一トポロジ IPv6

単一トポロジ IPv6 により、インターフェイス上で IPv4 ネットワーク プロトコルに加えて IPv6 用の IS-IS を設定できます。すべてのインターフェイスは同一のネットワーク プロトコル セットで構成されている必要があります。また、IS-IS エリア（レベル 1 ルーティング用）またはドメイン（レベル 2 ルーティング用）のすべてのルータは、すべてのインターフェイスで同一のネットワーク層プロトコル セットをサポートする必要があります。

single-topology モードでは、IPv6 トポロジは IPv4 ユニキャスト トポロジのナロー、ワイド メトリック スタイルの両方で機能します。single-topology での動作中は、レベルごとに 1 つの Shortest Path First（SPF）の計算が IPv4 ルートと IPv6 ルートの両方の計算に使用されます。IPv4 IS-IS と IPv6 IS-IS のルーティング プロトコルが共通のリンク トポロジを共有するため、単一の SPF の使用が可能です。

IS-IS のシングル トポロジの設定

IS-IS インスタンスをイネーブルにした後で、特定のネットワーク トポロジのルートを計算するように設定する必要があります。

ここでは、IPv4 または IPv6 トポロジ向けのインターフェイスで IS-IS プロトコルの動作を設定する方法について説明します。

始める前に

（注） ルータを single-topology モードで実行できるようにするには、IS-IS の各インターフェイスですべてのアドレス ファミリをイネーブルに設定し、IS-IS ルータ スタンザの IPv6 ユニキャスト アドレスファミリ内で「single-topology」を設定します。IPv6 アドレス ファミリ、または IPv4 と IPv6 の両方のアドレス ファミリを使用できますが、設定ではルータ上のすべてのアクティブなアドレス ファミリ セットを表します。さらに IPv6 ルータ アドレス ファミリ サブモードで single-topology を設定して、明示的に single-topology 動作をイネーブルにします。

これらの手順には例外が 2 つあります。

1. IS-IS プロセスの address-family スタンザに adjacency-check disable コマンドが含まれる場合、インターフェイスでアドレス ファミリをイネーブルにする必要はありません。
2. single-topology コマンドは ipv4 アドレス ファミリ サブモードでは無効です。

シングル トポロジのデフォルトのメトリック スタイルはナロー メトリックです。ワイド メトリックまたはナロー メトリックのどちらかを使用できます。この設定方法はシングル トポロジの設定に依存します。IPv4 と IPv6 の両方がイネーブルでシングル トポロジが設定されている場合には、メトリック スタイルは **address-family ipv4** スタンザ内で設定します。メトリックは **address-family ipv6** スタンザ内でも設定できますが、この場合には設定は無視されます。IPv6 のみがイネーブルでシングル トポロジが設定されている場合には、メトリック スタイルは **address-family ipv6** スタンザ内で設定します。

手順

ステップ 1 **configure**

ステップ 2 **interface** *type interface-path-id*

例：

```
RP/0/RP0/cpu 0: router(config)# interface HundredGigE 0/9/0/0
```

インターフェイス コンフィギュレーション モードを開始します。

ステップ 3 次のいずれかを実行します。

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

例：

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

または

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

または

インターフェイスの IPv4 アドレスを定義します。インターフェイスのいずれかで IS-IS ルーティングが設定されている場合は、IS-IS がイネーブルになっているエリアに含まれるすべてのインターフェイスで IP アドレスが必要です。

または

インターフェイスに割り当てられた IPv6 ネットワークを指定し、**eui-64** キーワードでインターフェイスの IPv6 処理をイネーブルにします。

または

インターフェイスに割り当てられた IPv6 インターフェイスを指定し、**link-local** キーワードでインターフェイスの IPv6 処理をイネーブルにします。

または

インターフェイスで IPv6 リンクローカル アドレスを自動的に設定し、インターフェイスで IPv6 処理もイネーブルにします。

- リンクローカル アドレスは、同じリンク上のノードとの通信にだけ使用できます。
- ipv6 address ipv6-prefix / prefix-length インターフェイス コンフィギュレーション コマンドを **eui-64** キーワードを付けずに指定すると、サイトローカルのグローバル IPv6 アドレスが設定されます。
- ipv6 address ipv6-prefix / prefix-length コマンドを **eui-64** キーワードとともに指定すると、IPv6 アドレスの下位 64 ビットにインターフェイス ID を持つサイトローカルのグローバル IPv6 アドレスが設定されます。指定する必要があるのはアドレスの 64 ビット ネットワーク プレフィックスだけです。最後の 64 ビットはインターフェイス ID から自動的に計算されます。
- ipv6 address コマンドを **link-local** キーワードとともに指定すると、IPv6 がインターフェイスでイネーブルになっている場合に自動的に設定されるリンクローカルアドレスの代わりに使用されるリンクローカル アドレスがインターフェイスに設定されます。

ステップ 4 **exit**

例：

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

インターフェイス コンフィギュレーション モードを終了し、ルータを モードに戻します。

ステップ 5 **router isis** *instance-id*

例：

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

指定したルーティング インスタンスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

- デフォルトでは、すべての IS-IS インスタンスがレベル 1 とレベル 2 になります。**is-type** コマンドを使用して、特定のルーティング インスタンスによって実行されるルーティングのレベルを変更できます。

ステップ 6 **net** *network-entity-title*

例：

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

ルーティング インスタンスの NET を設定します。

- マルチインスタンスの IS-IS を設定する場合は、ルーティング インスタンスごとに NET を指定します。NET およびアドレスの名前を指定できます。
- この例では、エリア ID が 47.0004.004d.0001 でシステム ID が 0001.0c11.1110.00 のルータを設定します。
- 複数のエリア アドレスを指定するには、追加の NET を指定します。NET のエリア アドレス部分が異なる場合でも、システム ID 部分はすべての設定項目で完全に一致する必要があります。

ステップ 7 **address-family ipv6** [**unicast**]

例：

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

IPv6 アドレス ファミリを指定し、ルータ アドレス ファミリ コンフィギュレーション モードを開始します。

- この例では、ユニキャスト IPv6 アドレス ファミリを指定します。

ステップ 8 **single-topology**

例：

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

（任意）IPv6 が設定されているときに IPv4 のリンク トポロジを設定します。

- **single-topology** コマンドは IPv6 サブモードでのみ有効です。このコマンドは、マルチトポロジ モードでデフォルトの設定である分離されたトポロジではなく、シングル トポロジを使用することを IPv6 に指示します。

ステップ 9 exit

例：

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

ルータ アドレス ファミリ コンフィギュレーション モードを終了して、ルータをルータ コンフィギュレーション モードに戻します。

ステップ 10 **interface** *type interface-path-id*

例：

```
RP/0/RP0/cpu 0: router(config-isis)# interface HundredGigE 0/9/0/0
```

インターフェイス コンフィギュレーション モードを開始します。

ステップ 11 **circuit-type** { **level-1** | **level-1-2** | **level-2-only** }

例：

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

（任意）隣接関係のタイプを設定します。

- デフォルトの回路タイプは設定済みの（**is-type** コマンドで設定した）システム タイプです。
- 通常、ルータを **level-1-2** のみとして設定し、**level-1** または **level-2-only** のみの隣接関係を形成するようにインターフェイスを制約する場合は、回線タイプを設定する必要があります。

ステップ 12 **address-family** { **ipv4** | **ipv6** } [**unicast**]

例：

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

IPv4 または IPv6 アドレス ファミリを指定して、インターフェイス アドレス ファミリ コンフィギュレーション モードを開始します。

- この例では、インターフェイスにユニキャスト IPv4 アドレス ファミリを指定します。

ステップ 13 **commit**

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

例：

```
RP/0/RP0/cpu 0: router# show isis interface HundredGigE 0/9/0/0
```

（任意）IS-IS インターフェイスに関する情報を表示します。

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

例：

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

（任意）すべてのエリアの接続済みルータのリストを表示します。

シングルトポロジ IS-IS for IPv6 の設定：例

次に、single-topology モードのイネーブル化の例を示します。IS-IS インスタンスが作成され、NET が定義され、インターフェイス上で IPv6 が IPv4 とともに設定され、IPv4 リンク トポロジが IPv6 で使用されます。この設定は、POS インターフェイス 0/3/0/0 が IPv4 アドレスと IPv6 アドレスの両方の隣接関係を形成できるようにします。

```
router isis isp
 net 49.0000.0000.0001.00
 address-family ipv6 unicast
  single-topology
 interface tenGigE 0/11/0/0
  address-family ipv4 unicast
  !
  address-family ipv6 unicast
  !
  exit
!
interface tenGigE 0/11/0/0
 ipv4 address 10.0.1.3 255.255.255.0
 ipv6 address 2001::1/64
```

単一トポロジ構成用の SPF 間隔の設定

ここでは、ルータのパフォーマンスをチューニングするために SPF 計算を調整する方法について説明します。このタスクはオプションです。

SPF 計算は特定のトポロジのルートを計算するため、チューニング属性はルータ アドレス ファミリ コンフィギュレーション サブモード内にあります。SPF 計算は、レベル 1 とレベル 2 のルートを別個に計算します。

IPv4 と IPv6 のアドレス ファミリが single-topology モードで使用される場合には、IPv4 トポロジ用の 1 つの SPF だけが存在します。IPv6 トポロジは IPv4 のトポロジを「借用」するため、IPv6 用の SPF 計算は必要ありません。single-topology モードで SPF 計算のパラメータを調整するには、address-family ipv4 unicast コマンドを設定します。

Incremental SPF アルゴリズムは、個別にイネーブルにできます。Incremental Shortest Path First（ISPF）は、イネーブルにしたときにすぐには適用されません。代わりにフル SPF アルゴリズムが使用されて、ISPF の実行に必要な状態情報の「シード」が作成されます。起動遅延により、IS-IS 再起動後の ISPF の実行が指定された期間止められます（データベースを安定させるため）。起動遅延期間が経過した後は、ISPF がすべての SPF 計算の実行について主要な役割を担います。シード更新間隔は、iSFP の状態の同期を維持するためにフル SPF の定期的な実行を可能にします。

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング インスタンスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

- **is-type** ルータ コンフィギュレーション コマンドを使用して、特定のルーティング インスタンスによって実行されるルーティングのレベルを変更できます。

ステップ 3 **address-family** { **ipv4** | **ipv6** } [**unicast**]

例：

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

IPv4 または IPv6 アドレス ファミリを指定して、ルータ アドレス ファミリ コンフィギュレーション モードを開始します。

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

例：

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

（任意）連続する SPF 計算の最小間隔を制御します。

- この値は、イベントがトリガーされた後の SPF 計算を遅延させ、SPF の実行の間に最小経過時間を適用させます。
- 小さすぎる値が設定された場合には、ネットワークが不安定なときにルータが大量の CPU リソースを失う可能性があります。
- 大きすぎる値が設定された場合には、ネットワーク トポロジの変更が遅延し、パケットを損失する結果になります。
- ISPF アルゴリズムは変更された LSP を受信するたびにすぐ実行されるため、SPF 間隔は ISPF の実行には適用されません。

ステップ 5 **ispf** [**level** { **1** | **2** }]

例：

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

（任意）Incremental IS-IS ISPF がネットワーク トポロジを計算するように設定します。

ステップ 6 **commit**

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

例：

```
RP/0/RP0/cpu 0: router# show isis instance 1 spf-log ipv4
```

（任意）ルータがフル SPF 計算を実行した頻度と、実行理由を表示します。

IS-IS のルートのカスタマイズ

ここでは、ルート機能を実行する方法について説明します。デフォルト ルートを IS-IS ルーティング ドメインに挿入する機能や別の IS-IS インスタンスで学習されたルートを再配布する機能が含まれます。このタスクはオプションです。

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング プロセスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

- デフォルトでは、すべての IS-IS インスタンスが自動的にレベル 1 とレベル 2 になります。**is-type** コマンドを使用して、特定のルーティング インスタンスによって実行されるルーティングのレベルを変更できます。

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

例：

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

（任意）過負荷ビットを設定します。

（注） NSF 再起動が再起動中に過負荷ビットを設定しないため、設定された過負荷ビットの動作は NSF の再起動に適用されません。

ステップ 4 **address-family** { **ipv4** | **ipv6** } [**unicast**]

例：

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

IPv4 または IPv6 アドレス ファミリを指定して、ルータ アドレス ファミリ コンフィギュレーション モードを開始します。

ステップ 5 **default-information originate** [**route-policy** *route-policy-name*]

例：

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

（任意）IPv4 または IPv6 のデフォルト ルートを IS-IS ルーティング ドメインに挿入します。

- **route-policy** キーワードと *route-policy-name* 引数により、IPv4 または IPv6 のデフォルト ルートをアドバタイズする条件を指定します。
- **route-policy** キーワードを省略すると、IPv4 または IPv6 のデフォルト ルートは無条件にレベル 2 でアドバタイズされます。

ステップ 6 **distribute-list** { {**prefix-list** *prefix-list-name* | **route-policy** *route-policy-name*} } **in**

例：

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

（任意）Intermediate System-to-Intermediate System（IS-IS）がルーティング情報ベース（RIB）にインストールするルートをフィルタリングします。

警告 **distribute-list in** コマンドが設定されている場合、IS-IS で計算される一部のルートはローカル ルータのフォワーディング プレーンにインストールされませんが、他の IS-IS ルータはこれを認識しません。このため、他の IS-IS ルータで計算されたフォワーディング ステートとこのルータ上の実際のフォワーディング ステートに違いが生まれます。場合によっては、トラフィックがドロップまたはループする可能性があります。このため、このコマンドを使用するタイミングに注意してください。

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

例：

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

（任意）ある IS-IS インスタンスから別のインスタンスにルートを再配布します。

- この例では、IS-IS インスタンスは別の IS-IS インスタンスからのレベル 1 ルートを再配布します。

ステップ 8 次のいずれかを実行します。

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

例：

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

または

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

（任意）レベル 1-2 ルータがサマリーをアドバタイズするときに直接レベル 1 プレフィックスをアドバタイズするのではなく、レベル 1 IPv4 および IPv6 プレフィックスをレベル 2 で集約できるようにします。

- この例では、IPv4 アドレスおよびマスクを指定します。

または

- この例では IPv6 プレフィックスを指定し、コマンドは RFC 2373 に記載された形式にする必要があり、16 ビット値をコロンで区切った 16 進でアドレスを指定します。
- IPv6 プレフィックスは IPv6 ルータ アドレス ファミリ コンフィギュレーション サブモードでのみ設定でき、IPv4 プレフィックスは IPv4 ルータ アドレス ファミリ コンフィギュレーション サブモードでのみ設定できます。

ステップ 9 **maximum-paths** *route-number*

例：

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

（任意）ルーティング テーブルで許可されるパラレル パスの最大数を設定します。

ステップ 10 **distance** *weight* [*address / prefix-length* [*route-list-name*]]

例：

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

（任意）IS-IS プロトコルにより発見されたルートに割り当てられるアドミニストレーティブ ディスタンスを定義します。

- IPv4 と IPv6 で異なるアドミニストレーティブ ディスタンスを適用できます。

ステップ 11 **set-attached-bit**

例：

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

（任意）IS-IS インスタンスにレベル 1 LSP 内の attached ビットを設定します。

ステップ 12 **commit**

複数インスタンス間での IS-IS ルートの再配布：例

次に、set-attached-bit および redistribute コマンドの使用例を示します。レベル 1 に制限されたインスタンス「1」とレベル 2 に制限されたインスタンス「2」の 2 つのインスタンスが設定されています。

再配布を使用してレベル 1 のインスタンスからレベル 2 のインスタンスにルートが伝播します。レベル 1 のルートが優先されるように、レベル 2 インスタンスのアドミニストレーティブ ディスタンスが明示的に大きく設定されていることに注目してください。

レベル 1 インスタンスはレベル 2 インスタンスへの再配布ルートであることから、レベル 1 インスタンスには attached ビットが設定されています。このため、インスタンス「1」はエリアからバックボーンへ到達するための適切な候補になります。

```
  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 0/9/0/0
   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 0/9/0/0
   address-family ipv4 unicast
```

RIB にプレフィックスを追加するためのプライオリティの設定

このオプションの手順では、指定されたプレフィックスを RIB に追加するプライオリティ（順序）の設定方法について説明します。プレフィックスは、アクセスリスト（ACL）、プレフィックス リスト、またはタグ値の照合を使用して選択できます。

手順

ステップ 1　configure

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング プロセスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。この例では、IS-IS インスタンスは isp と呼ばれます。

ステップ 3 **address-family** { **ipv4** | **ipv6** } [**unicast**]

例：

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

IPv4 または IPv6 アドレス ファミリを指定して、ルータ アドレス ファミリ コンフィギュレーション モードを開始します。

ステップ 4 **metric-style wide** [**transition**] [**level** { **1** | **2** }]

例：

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

レベル 1 エリアでワイド リンク メトリックのみを生成して受け入れるようにルータを設定します。

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

例：

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

値が 3 のタグが付けられたすべてのルートを先にインストールします。

ステップ 6 **commit**

IS-IS のインターフェイス

IS-IS のインターフェイスは次のタイプのいずれかとして設定できます。

- アクティブ：接続されたプレフィックスをアドバタイズし、隣接関係を形成します。これはデフォルトのインターフェイスです。
- パッシブ：接続されたプレフィックスをアドバタイズしますが、隣接関係は形成しません。インターフェイスをパッシブに設定するには、passive コマンドを使用します。パッシブなインターフェイスは、IS-IS ドメインへの挿入が必要なループバック アドレスのような、重要なプレフィックスのために控えめに使用します。多くの接続されたプレフィックスをアドバタイズする必要がある場合には、適切なポリシーを備えた接続ルートの再配布を代わりに使用します。

- 抑制：接続されたプレフィックスをアドバタイズせず、隣接関係を形成します。インターフェイスを抑制に設定するには、suppress コマンドを使用します。
- シャットダウン：接続されたプレフィックスをアドバタイズせず、隣接関係も形成しません。IS-IS の設定を削除せずにインターフェイスをディセーブルにするには、shutdown コマンドを使用します。

IS-IS インターフェイス ルートのタギング

このオプションの手順では、IS-IS インターフェイスの接続されたルートにタグを関連付ける方法について説明します。

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング プロセスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。この例では、IS-IS インスタンスは isp と呼ばれます。

ステップ 3 **address-family** { **ipv4** | **ipv6** } [**unicast**]

例：

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

IPv4 または IPv6 アドレス ファミリを指定して、ルータ アドレス ファミリ コンフィギュレーション モードを開始します。

ステップ 4 **metric-style wide** [**transition**] [**level** { **1** | **2** }]

例：

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

レベル 1 エリアでワイド リンク メトリックのみを生成して受け入れるようにルータを設定します。

ステップ 5 **exit**

例：

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

ルータ アドレス ファミリ コンフィギュレーション モードを終了して、ルータをルータ コンフィギュレーション モードに戻します。

ステップ 6 **interface** *type number*

例：

```
RP/0/RP0/cpu 0: router(config-isis)# interface HundredGigE 0/9/0/0
```

インターフェイス コンフィギュレーション モードを開始します。

ステップ 7 **address-family** { **ipv4** | **ipv6** } [**unicast**]

例：

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

IPv4 または IPv6 アドレス ファミリを指定して、アドレス ファミリ コンフィギュレーション モードを開始します。

ステップ 8 **tag** *tag*

例：

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

アドバタイズされた接続されたルートに関連付けるタグの値を設定します。

ステップ 9 **commit**

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

例：

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

タグ情報を表示します。すべてのタグが RIB に存在することを確認します。

ルートのタギング：例

次に、ルートのタギングの例を示します。

```
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
```

LSP フラッディングの制限

リンク ステート パケット（LSP）を制限すると、特定の「メッシュの」ネットワーク トポロジで有効な場合があります。このようなネットワークの例は、非ブロードキャストマルチアクセス（NBMA）トランスポート上の完全メッシュ化されたポイントツーポイントリンクのセットなどの冗長性の高いネットワークです。このようなネットワークでは、完全な LSP フラッディングにより、ネットワークのスケーラビリティを制限できます。フラッディングのドメインのサイズを制限する 1 つの方法は、複数のレベル 1 エリアと 1 つのレベル 2 エリアを使用することにより、階層を導入することです。ただし、階層の代わりに他の 2 つの技法を使用することもできます。特定のインターフェイス上でフラッディングをブロックし、メッシュ グループを設定します。

両方の技法は、LSP フラッディングを何らかの方法で制限することで動作します。直接的な結果として、ネットワークのスケーラビリティが改善される一方で、ネットワークの（障害時の）信頼性が低下します。ブロッキングやメッシュ グループによって使用が制限されていない場合、フラッディングが可能なリンクが存在しても、一連の障害によって LSP をネットワーク全体にフラッディングできないことがあるからです。このような場合、ネットワーク内の異なるルータのリンク ステート データベースを、同期できないことがあります。永続的な転送ループのような問題が結果として発生する可能性があります。したがって、ブロッキングやメッシュ グループはどうしても必要な場合にかぎり、慎重にネットワークを設計したうえで使用することを推奨します。

IS-IS の LSP フラッディングの制御

LSP フラッディングにより、ネットワークのスケーラビリティを制限できます。ルータでグローバルに、またはインターフェイスで LSP データベース パラメータを調整することによって、LSP フラッディングを制御できます。このタスクはオプションです。

LSP フラッディングを制御するコマンドの多くには、適用されるレベルを指定するオプションが含まれます。オプションを指定しなかった場合、コマンドは両方のレベルに適用されます。オプションが 1 つのレベルに設定された場合、もう一方のレベルはデフォルト値を使用し続けます。両方のレベルのオプションを設定するには、コマンド twice を使用します。次に例を示します。

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

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング インスタンスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

- **is-type** ルータ コンフィギュレーション コマンドを使用して、特定のルーティング インスタンスによって実行されるルーティングのレベルを変更できます。

ステップ 3 **lsp-refresh-interval** *seconds* [**level** { **1** | **2** }]

例：

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

（任意）異なるシーケンス番号を持つ LSP を再生成する間隔を設定します。

- 更新間隔は、常に、max-lsp-lifetime コマンドよりも低く設定する必要があります。

ステップ 4 **lsp-check-interval** *seconds* [**level** { **1** | **2** }]

例：

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

（任意）データベースの LSP のチェックサムを検証するデータベース全体の定期チェックの間隔を設定します。

- この操作は、CPU の点でコスト高であるため、あまり発生しないように設定する必要があります。

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

例：

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

（任意）ネットワークが不安定な間は LSP の生成レートを低下させます。ルータの CPU 負荷を軽減し、IS-IS ネイバーへの LSP 送信数を低減するのに役立ちます。

- ネットワークの不安定性が長引いている間に LSP の再計算を繰り返すと、ローカル ルータの CPU 負荷が増加する可能性があります。さらに、これらの再計算された LSP をネッ

トワーク内の他の中継システムにフラッディングすると、トラフィックが増加し、他のルータがルート計算を実行するために費やす時間が増加する可能性があります。

ステップ 6 **lsp-mtu** *bytes* [**level** { **1** | **2** }]

例：

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

（任意）LSP の最大伝送単位（MTU）サイズを設定します。

ステップ 7 **max-lsp-lifetime** *seconds* [**level** { **1** | **2** }]

例：

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

（任意）ルータから発信された LSP に設定する最初のライフタイムを設定します。

- これは、LSPが再生成または更新されない場合に、ネイバーのデータベースにLSPが維持される時間です。

ステップ 8 **ignore-lsp-errors disable**

例：

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

（任意）チェックサム エラーで受信した LSP をパージするようにルータを設定します。

ステップ 9 **interface** *type interface-path-id*

例：

```
RP/0/RP0/cpu 0: router(config-isis)# interface HundredGigE 0/9/0/0
```

インターフェイス コンフィギュレーション モードを開始します。

ステップ 10 **lsp-interval** *milliseconds* [**level** { **1** | **2** }]

例：

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

（任意）インターフェイス上で送信された各 LSP 間の時間を設定します。

ステップ 11 **csnp-interval** *seconds* [**level** { **1** | **2** }]

例：

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

（任意）ブロードキャストインターフェイス上で定期的にCSNPパケットが送信される間隔を設定します。

- より頻繁にCSNPを送信することは、受信のために隣接ルータはより激しく動作する必要があることを意味します。
- CSNP の送信の頻度を下げることは、隣接ルータ間の相違がより長く続くことを意味します。

ステップ 12 **retransmit-interval** *seconds* [**level** { **1** | **2** }]

例：

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

（任意）LSPが受信されていないと判断して再送信するまでに送信ルータが応答を待つ時間を設定します。

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

ステップ 13 **retransmit-throttle-interval** *milliseconds* [**level** { **1** | **2** }]

例：

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

（任意）ポイントツーポイント インターフェイス上の各 LSP の再送信間隔を設定します。

- この時間は通常 lsp-interval コマンドの時間以上にします。これは隣接ルータがビジーであることがLSPが失われた原因の可能性があるためです。間隔を長くするとネイバーはより時間をかけて送信を受け取ることができます。

ステップ 14 **mesh-group** { *number* | **blocked** }

例：

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

（任意）NBMAネットワークのLSPフラッディングを、高度にメッシュ化されたポイントツーポイント トポロジで最適化します。

- このコマンドは高度にメッシュ化されたポイントツーポイント トポロジの NBMA ネットワークのみに適しています。

ステップ 15 **commit**

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

例：

```
RP/0/RP0/cpu 0: router# show isis interface HundredGigE 0/9/0/0 brief
```

（任意）IS-IS インターフェイスに関する情報を表示します。

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

例：

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

（任意）IS-IS LSP データベースを表示します。

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

例：

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

（任意）LSP ログ情報を表示します。

ステップ 19 **show isis database-log** [**level** { **1** | **2** }]

例：

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

（任意）IS-IS データベース ログ情報を表示します。

残りの最小ライフタイム

残りの最小ライフタイム機能は、LSPの早期消去と不要なフラッディングを防止します。残りのライフタイムフィールドがフラッディング中に破損した場合、この破損は検出されません。このような破損の結果は、残りのライフタイム値がどのように変更されるかによって異なります。この機能により、IS-IS で受信した LSP の残りのライフタイム値を LSP 最大ライフタイムにリセットすることで、この問題が解決されます。デフォルトでは、LSP最大ライフタイムは 1200 秒に設定され、**max-lsp-lifetime** *seconds* コマンドを使用すると異なる値を設定できます。このアクションによって、受信した残りのライフタイムがどんな値であっても、LSP最大ライフタイムまで LSP がデータベース内に存在する限り、LSP の発信元以外のシステムが LSP を消去しないことが保障されます。

LSP の残りのライフタイムが 0 に達すると、LSP はリンク ステート データベースでさらに 60 秒間保持されます。この追加のライフタイムはゼロ エージング ライフタイムと呼ばれます。ゼロ エージング ライフタイムが経過しても、対応するルータが LSP を更新しない場合、LSP はリンク ステート データベースから削除されます。

また、残りのライフタイムフィールドはネットワークの問題を特定する場合にも役立ちます。受信した LSP のライフタイム値がゼロ エージング ライフタイム（60 秒）未満の場合、IS-IS は破損したライフタイム イベントであることを示すエラー メッセージを生成します。エラー メッセージの例は次のとおりです。

```
Dec 14 15:36:45.663 : isis[1011]: RECV L2 LSP 1111.1111.1112.03-00 from 1111.1111.1112.03:

 possible corrupted lifetime 59 secs for L2 lsp 1111.1111.1112.03-00 from SNPA
02e9.4522.5326 detected.
```

IS-IS は LSP データベースに受信した残りのライフタイム値を保存します。値は、**Rcvd** フィールド内の **show isis database** コマンド出力に表示されます。

IS-IS 認証

隣接関係の確立を制限するために、hello-password コマンドを使用して認証ができます。また、LSP の交換を制限するために、lsp-password コマンドを使用して認証ができます。

IS-IS はプレーン テキスト認証をサポートしますが、この認証は、無許可のユーザに対するセキュリティを提供しません。プレーンテキスト認証ではパスワードが設定でき、無許可のネットワーキング デバイスがルータと隣接関係を形成することを防ぐことができます。このパスワードはプレーンテキストで交換されるため、IS-IS パケットを表示できるエージェントによって参照される可能性があります。

HMAC-MD5 パスワードが設定されている場合、パスワードはネットワークを介して送信されず、代わりに交換データの完全性を確認するための暗号化チェックサムを計算するために使用されます。

IS-IS では、設定されたパスワードを単純な暗号を使用して保存します。ただし、プレーン テキスト形式のパスワードが、LSP、Sequence Number Protocol（SNP）、hello パケットで使用され、IS-IS パケットを表示するプロセスによって参照される可能性があります。パスワードはプレーン テキスト（クリア テキスト）形式または暗号化形式で入力できます。

ドメイン パスワードを設定するには、レベル 2 で lsp-password コマンドを設定します。エリア パスワードを設定するには、レベル 1 で lsp-password コマンドを設定します。

キーチェーン機能によって、IS-IS で設定済みのキーチェーンを参照できます。IS-IS キーチェーンは、hello および LSP のキーチェーン認証をイネーブルにします。キーチェーンは、IS-IS 内のルータ レベル（lsp-password コマンドの場合）およびインターフェイス レベル（hello-password コマンドの場合）で設定できます。これらのコマンドでは、グローバル キーチェーン設定を参照して、設定されているキーチェーンのグローバル セットからセキュリティ パラメータを取得するように IS-IS プロトコルに指示します。

IS-IS はキーチェーンを使用して、認証のためにヒットレス キー ロールオーバーを実装できます。キー ロールオーバーの仕様は時間に基づき、ピア間に時計の誤差が発生すると、ロールオーバープロセスが影響を受けます。許容値の指定を設定できるため、承認時間枠をその分だけ（前後に）拡張できます。この承認時間枠により、アプリケーション（ルーティング プロトコルおよび管理プロトコルなど）のヒットレス キー ロールオーバーが容易になります。

IS-IS の認証の設定

ここでは、IS-IS の認証の設定方法について説明します。このタスクはオプションです。

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング インスタンスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

- **is-type** コマンドを使用して、特定のルーティング インスタンスによって実行されるルーティングのレベルを変更できます。

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

例：

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

LSP 認証パスワードを設定します。

- **hmac-md5** キーワードは、パスワードが HMAC-MD5 認証で使用されることを指定します。
- **text** キーワードは、パスワードがクリアテキスト パスワード認証で使用されることを指定します。
- **clear** キーワードは、入力時にパスワードが暗号化されないことを指定します。
- **encrypted** キーワードは、パスワードが入力時に双方向アルゴリズムを使用して暗号化されていることを指定します。
- **level 1** キーワードは、エリア内の認証のパスワードを設定します（レベル 1 LSP と SNP レベル）。
- **level 2** キーワードは、バックボーン（レベル 2 エリア）の認証パスワードを設定します。
- **send-only** キーワードは、LSP とシーケンス番号プロトコル データ ユニット（SNP）の送信時にこれらに認証を追加します。受信 LSP または SNP は認証されません。
- **snp send-only** キーワードは SNP の送信時に SNP に認証を追加します。受信 SNP は認証されません。

（注） SNP パスワード チェックをディセーブルにするには、 **snp send-only** キーワードを **lsp-password** コマンドで指定する必要があります。

ステップ 4 **interface** *type interface-path-id*

例：

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

インターフェイス コンフィギュレーション モードを開始します。

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

例：

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

IS-IS インターフェイスの認証パスワードを設定します。

ステップ 6 **commit**

IS-IS のキーチェーンの設定

ここでは、IS-IS のキーチェーンの設定方法について説明します。このタスクはオプションです。

キーチェーンは IS-IS 内のルータ レベル（**lsp-password** コマンド）およびインターフェイス レベル（**hello-password** コマンド）で設定できます。これらのコマンドでは、グローバル キーチェーン設定を参照して、設定されているキーチェーンのグローバルセットからセキュリティ パラメータを取得するように IS-IS プロトコルに指示します。ルータ レベルの設定（lsp-password コマンド）では、ルータで生成されるすべての IS-IS LSP と、すべての Sequence Number Protocol Data Unit（SN PDU）でキーチェーンを使用するように設定します。HELLO PDU で使用されるキーチェーンはインターフェイス レベルで設定され、IS-IS が設定されたインターフェイスごとに異なる値を設定できます。

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング インスタンスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

- **is-type** コマンドを使用して、特定のルーティング インスタンスによって実行されるルーティングのレベルを変更できます。

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

例：

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

キーチェーンを設定します。

ステップ 4 **interface** *type interface-path-id*

例：

```
RP/0/RP0/cpu 0: router(config-isis)# interface HundredGigE 0/9/0/0
```

インターフェイス コンフィギュレーション モードを開始します。

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

例：

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

IS-IS インターフェイスの認証パスワードを設定します。

ステップ 6 **commit**

ノンストップ フォワーディング

Cisco IOS XR ソフトウェア では、IS-IS NSF により IS-IS プロセスの再起動後にユーザがネットワークにアクセスできない時間が最小限になります。

IS-IS プロセスが再起動すると、そのデバイスのすべてのルーティング ピアは、デバイスがダウンし、その後再びアップになったことを検知します。このような移行によって、いわゆるルーティング フラップが発生します。ルーティング フラップは、複数のルーティング ドメインに広がる場合があります。ルーティングの再起動によって発生したルーティング フラップによって、ルーティングが不安定になります。これはネットワーク全体のパフォーマンスに悪影響を及ぼします。NSF はルーティング フラップを抑止することによって、ネットワークの安定性を保ちます。

NSF では、プロセスの再起動後にルーティング プロトコル情報を復元する一方で、データ パケットの転送を既知のルートで継続できます。NSF 機能が設定されると、ピア ネットワーキング デバイスではルーティング フラップが発生しません。RP フェールオーバー イベント間のルーティングを維持するには、NSF に加えて NSR を設定する必要があります。

IS-IS ルーティングを実行している Cisco IOS XR ルータがプロセスの再起動を行うときは、リンクステート データベースを IS-IS ネイバーと再同期するために、2 つのタスクをルータが実行する必要があります。まず、ネイバー関係をリセットせずに、ネットワーク上の使用可能な IS-IS ネイバーを再学習します。次に、ルータはネットワークのリンクステート データベースのコンテンツを再取得します。

NSF を設定する場合、IS-IS NSF 機能には次の 2 つのオプションがあります。

- IETF NSF
- Cisco NSF

ネットワーク セグメント上の隣接ルータが NSF 対応の場合、つまり隣接ルータが RFC5306 をサポートするソフトウェア バージョンを実行している場合、それらのルータは、**nsf ietf** コマンドで設定されたルータの再起動をサポートします。IETF NSF を使用すると、隣接ルータは、

フェールオーバー後のルーティング情報を再構築するための隣接情報およびリンク ステート情報を提供します。

Cisco IOS XR ソフトウェア では、Cisco NSF が再起動からの回復に必要なすべての状態をチェックポイントで（永続的に）保存し、隣接ルータからの特別な協力を必要としません。状態は隣接ルータによって回復されますが、IS-IS ルーティング プロトコルの標準機能のみを使用します。この機能により Cisco NSF は、他のルータが IETF 標準の NSF 実装を使用していないネットワークでの使用に適しています。

（注） IETF NSF を Cisco IOS XR ルータで設定し、隣接ルータが IETF NSF をサポートしていない場合には、隣接はフラップの影響を受けますが、IETF NSF をサポートしているすべてのネイバーではノンストップ フォワーディングが維持されます。IETF NSF をサポートするネイバーがない場合は、再起動はコールド スタートになります。

IS-IS のノンストップ フォワーディングの設定

ここでは、ルータに NSF を設定する方法について説明します。NSF は、ソフトウェアがプロセスの再起動後に IS-IS リンクステート データベースを IS-IS ネイバーと再同期できるようにします。プロセスは次の原因で再起動する可能性があります。

- RP フェールオーバー（ウォーム リスタートのため）
- 単純なプロセスの再起動（IS-IS のリロードなどの管理要求によるプロセスの再起動）
- IS-IS のソフトウェア アップグレード

いずれの場合でも、NSF はリンク フラップおよびユーザ セッションの損失を低減します。このタスクはオプションです。

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング インスタンスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

- **is-type** ルータ コンフィギュレーション コマンドを使用して、特定のルーティング インスタンスによって実行されるルーティングのレベルを変更できます。

ステップ 3 **nsf** { **cisco** | **ietf** }

例：

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

次の再起動で NSF をイネーブルにします。

- NSF 対応ネットワーキング デバイスが隣接していない可能性がある異種ネットワークで IS-IS を実行するには、 **cisco** キーワードを入力します。
- 隣接するすべてのネットワーキング デバイスが IETF ドラフトベースの再起動性をサポートする同種ネットワークで IS-IS をイネーブルにするには、**ietf** キーワードを入力します。

ステップ 4 **nsf interface-expires** *number*

例：

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

確認された NSF の再開確認応答を再送信する回数を設定します。

- NSF 再起動の間に再送上限数に達した場合、再起動はコールド リスタートになります。

ステップ 5 **nsf interface-timer** *seconds*

例：

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

各再起動応答を待機する秒数を設定します。

ステップ 6 **nsf lifetime** *seconds*

例：

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

NSF 再開に続くルートの最大有効期間を設定します。

- この設定時間は再起動の最中にルーティング情報ベース（RIB）にルートを維持する時間であるため、このコマンドには NSF 再起動全体の実行に必要な時間を設定します。
- 設定する値が大きすぎると、古いルートが残ります。
- 設定する値が小さすぎると、ルートの破棄が早すぎる結果になります。

ステップ 7 **commit**

ステップ 8 **show running-config** [*command*]

例：

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

（任意）現在の実行コンフィギュレーション ファイルの内容全体またはファイルのサブセットを表示します。

- NSF 対応デバイスの IS-IS 設定に「nsf」が表示されていることを確認します。
- この例では、コンフィギュレーション ファイルの内容の「isp」インスタンスのみを示しています。

ISIS NSR

ノンストップ ルーティング（NSR）は、プロセッサのスイッチオーバー イベント（RP フェールオーバーまたは ISSU）中に冗長なルート プロセッサを持つデバイスの IS-IS ルーティングの変更を抑制し、ネットワークの不安定性とダウンタイムを低減します。ノンストップ ルーティングが使用されている場合、アクティブからスタンバイ RP への切り替えは、ネットワーク内の他の IS-IS ルータには影響しません。ルーティング プロトコルのピアリング状態を継続するのに必要なすべての情報は、スイッチオーバー前にスタンバイ プロセッサに転送されるため、スイッチオーバー直後に処理を続行できます。

プロセスの再起動間のルーティングを維持するには、NSR に加えて NSF を設定する必要があります。

ISIS-NSR の設定

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング インスタンスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

ステップ 3 **nsr**

例：

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

NSR 機能を設定します。

ステップ 4 **commit**

ステップ 5 **show isis nsr adjacency**

例：

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

隣接関係情報を表示します。

ステップ 6 **show isis nsr status**

例：

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

NSR のステータス情報を表示します。

ステップ 7 **show isis nsr statistics**

例：

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

アクティブ ルータおよびスタンバイ ルータ上の ISIS 隣接関係、lsp、ルート、トンネル、Te リンクの数を示します。

マルチプロトコル ラベル スイッチング トラフィック エンジニアリング

MPLS TE 機能を使用すると、MPLS バックボーンで、レイヤ 2 ATM およびフレーム リレー ネットワークが持つトラフィック エンジニアリングの能力を再現し、そのうえで機能を拡張することができます。MPLS は、レイヤ 2 テクノロジーとレイヤ 3 テクノロジーを統合したものです。

IS-IS では、MPLS TE はリソース予約プロトコル（RSVP）を使用して自動的にバックボーン全体に MPLS TE ラベルスイッチド パスを確立して維持します。ラベルスイッチド パスが使用するルートは、ラベルスイッチド パスのリソース要件とネットワーク リソース（帯域幅など）によって決定されます。利用可能なリソースは、IS-IS の特別な IS-IS TLV 拡張を使用してフラッディングされます。ラベルスイッチド パスは明示的なルートであり、トラフィック エンジニアリング（TE）トンネルとして参照されます。

IS-IS の MPLS トラフィック エンジニアリングの設定

このタスクでは、MPLS TE の IS-IS を設定する手順について説明します。このタスクはオプションです。

始める前に

ルータで IS-IS の MPLS TE をイネーブルにする前に、ネットワークで MPLS ソフトウェア機能をサポートする必要があります。

（注） ネットワークのトラフィック エンジニアリング部分にあるすべての IS-IS ルータ上で、次のタスク リストのコマンドを入力する必要があります。

（注） MPLS トラフィック エンジニアリングでは、現在、番号なし IP リンクを介したルーティングおよびシグナリングはサポートされていません。このため、このようなリンク上には、この機能を設定しないでください。

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング インスタンスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

- **is-type** ルータ コンフィギュレーション コマンドを使用して、特定のルーティング インスタンスによって実行されるルーティングのレベルを変更できます。

ステップ 3 **address-family** { **ipv4** | **ipv6** } [**unicast**]

例：

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

IPv4 または IPv6 アドレス ファミリを指定して、ルータ アドレス ファミリ コンフィギュレーション モードを開始します。

ステップ 4 **mpls traffic-eng level** { **1** | **2** }

例：

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

指定した IS-IS レベルに MPLS TE リンク情報をフラッディングするように IS-IS を実行するルータを設定します。

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

例：

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

ノードの MPLS TE ルータ ID を指定した IP アドレスまたは指定したインターフェイスに関連付けられている IP アドレスにするように指定します。

ステップ 6 **metric-style wide** [**level** { **1** | **2** }]

例：

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

レベル 1 エリアでワイド リンク メトリックのみを生成して受け入れるようにルータを設定します。

ステップ 7 **commit**

ステップ 8 **show isis** [**instance** *instance-id*] **mpls traffic-eng tunnel**

例：

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

（任意）MPLS TE トンネル情報を表示します。

ステップ 9 **show isis** [**instance** *instance-id*] **mpls traffic-eng adjacency-log**

例：

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

（任意）MPLS TE IS-IS 隣接変更のログを表示します。

ステップ 10 **show isis** [**instance** *instance-id*] **mpls traffic-eng advertisements**

例：

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

（任意）MPLS TE から最後にフラッディングされた記録を表示します。

MPLS TE 転送隣接

MPLS TE 転送隣接により、ネットワーク管理者はトラフィック エンジニアリングおよびラベル スイッチ パス（LSP）トンネルを、Shortest Path First（SPF）アルゴリズムに基づいた内部ゲートウェイ プロトコル（IGP）ネットワーク内のリンクとして処理できます。転送隣接は、同じ IS-IS レベルのルータ間で作成できます。ルータとルータは、間に何個かホップを入れて配置できます。この結果、TE トンネルに関連付けられたリンク コストで、IGP ネットワーク内のリンクとして、アドバタイズされます。TE ドメインの外側にあるルータは、TE トンネルを参照し、その TE トンネルを使用して、ネットワーク内でトラフィックをルーティングするための最短パスを計算します。

MPLS TE 転送隣接は、双方向接続性確認に成功した場合のみ IS-IS SPF で考慮されます。これには転送隣接が双方向であるか、または MPLS TE トンネルのヘッドエンドとテール エンドのルータが隣接している場合が該当します。

MPLS TE 転送隣接機能は、IS-IS でサポートされます。MPLS TE 転送隣接機能の設定の詳細については、『MPLS Configuration Guide』を参照してください。

IS-IS の隣接関係の調整

このタスクでは、隣接状態変更のロギングをイネーブルにし、IS-IS 隣接パケットのタイマーを変更して、隣接状態のさまざまな側面を表示する方法について説明します。IS-IS 隣接を調整することにより、リンクで輻輳が発生している場合のネットワークの安定性が向上します。このタスクはオプションです。

ポイントツーポイント リンクの場合、IS-IS はレベル 1 とレベル 2 に対して単一の hello だけを送信します。これは、ポイントツーポイント リンクでの level 修飾子が無意味であることを意味します。ポイントツーポイント インターフェイスの hello パラメータを変更するには、level オプションの指定を省略します。

インターフェイス サブモードで設定可能なオプションは、そのインターフェイスだけに適用されます。デフォルトで、値はレベル 1 とレベル 2 に適用されます。

hello-password コマンドを使用して無許可のルータや望ましくないルータとの隣接関係の形成を防ぐことができます。この機能は、隣接関係の確立が望ましくないルータとの接続が多く見られる LAN では特に有効です。

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング インスタンスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

- **is-type** コマンドを使用して、特定のルーティング インスタンスによって実行されるルーティングのレベルを変更できます。

ステップ 3 **log adjacency changes**

例：

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

IS-IS の隣接状態の変更時にログ メッセージを生成します（Up または Down）。

ステップ 4 **interface** *type interface-path-id*

例：

```
RP/0/RP0/cpu 0: router(config-isis)# interface HundredGigE 0/9/0/0
```

インターフェイス コンフィギュレーション モードを開始します。

ステップ 5 **hello-padding** { **disable** | **sometimes** } [**level** { **1** | **2** }]

例：

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

ルータの IS-IS インターフェイスの IS-IS hello PDU のパディングを設定します。

- helloパディングはこのインターフェイスのみに適用され、すべてのインターフェイスには適用されません。

ステップ 6 **hello-interval** *seconds* [**level** { **1** | **2** }]

例：

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

ソフトウェアが送信する hello パケット間の時間間隔を指定します。

ステップ 7 **hello-multiplier** *multiplier* [**level** { **1** | **2** }]

例：

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

ネイバーが見落とすことができる IS-IS hello パケット数の最大値を指定します。見落とされたパケット数がこの値を超えると、ルータは隣接がダウンしていると宣言します。

- 大きい値にするとネットワークが許容するドロップ パケットの数が増加しますが、隣接ルータの障害の検出に必要な時間も増加します。
- 隣接ルータの障害が検出されないと、逆により多くのパケットが失われる結果になる可能性があります。

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

例：

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

このシステムが hello パケットの認証を含むことを指定し、ネイバーからの hello パケットの認証が成功し、隣接関係が確立することが必要です。

ステップ 9 **commit**

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

例：

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

（任意）IS-IS 隣接を表示します。

ステップ 11 show isis adjacency-log

例：

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

（任意）最新の隣接状態の遷移ログを表示します。

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

例：

```
RP/0/RP0/cpu 0: router# show isis interface HundredGigE 0/9/0/0 brief
```

（任意）IS-IS インターフェイスに関する情報を表示します。

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

例：

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

（任意）IS-IS ネイバーに関する情報を表示します。

MPLS ラベル配布プロトコル IGP 同期

マルチ プロトコル ラベル スイッチング（MPLS）ラベル配布プロトコル（LDP）内部ゲートウェイ プロトコル（IGP）同期では、IGP パスをスイッチングが使用される前に LDP のラベル交換を完了させることができます。次の 2 つの状況では MPLS のトラフィック損失が発生する可能性があります。

- IGP 隣接が確立されると、LDP がそのリンク ピアとラベルを交換する前に、ルータが新しい隣接を使用してパケットの転送を開始します。
- LDP セッションが閉じられるときに、確立した LDP セッションの代替パスを使用せずに LDP ピアと関連付けられたリンクを使用してルータがトラフィックの転送を続ける。

この機能は、LDP と IS-IS を同期させるメカニズムを提供し、MPLS のパケット損失を最小化します。この同期は、LDP セッションの状態に基づいてネイバーの IS-IS リンクステート パケット（LSP）のリンク メトリックを変更することで実現されます。

リンク上で IS-IS の隣接関係は確立されているが、LDP セッションが失われているかまたは LDP がラベルの交換をまだ完了していないときには、IS-IS は最大のメトリックをそのリンクでアドバタイズします。このインスタンスでは、LDP IS-IS 同期はまだ実現されていません。

（注） IS-IS では、最大のメトリック（0xFFFFFF）を持つリンクは Shortest Path First（SPF）として考慮されません。このため最大のメトリックである -1（0XFFFFFE）が MPLS LDP IGP 同期で使用されます。

LDP IS-IS 同期が達成されると、IS-IS は通常の（設定されたまたはデフォルトの）メトリックをそのリンクでアドバタイズします。

MPLS LDP IS-IS 同期の設定

このタスクは、マルチプロトコルラベルスイッチング（MPLS）ラベル配布プロトコル（LDP）IS-IS 同期をイネーブルにする方法について説明します。MPLS LDP 同期は、インターフェイス コンフィギュレーション モードで、アドレス ファミリに対してイネーブルにすることができます。IPv4 ユニキャスト アドレス ファミリのみがサポートされます。このタスクはオプションです。

手順

ステップ 1 **configure**

ステップ 2 **router isis** *instance-id*

例：

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

指定したルーティング プロセスの IS-IS ルーティングをイネーブルにし、ルータをルータ コンフィギュレーション モードにします。

- デフォルトでは、すべての IS-IS インスタンスが自動的にレベル 1 とレベル 2 になります。**is-type** コマンドを使用して、特定のルーティング インスタンスによって実行されるルーティングのレベルを変更できます。

ステップ 3 **interface** *type interface-path-id*

例：

```
RP/0/RP0/cpu 0: router(config-isis)# interface HundredGigE 0/9/0/0
```

インターフェイス コンフィギュレーション モードを開始します。

ステップ 4 **address-family ipv4 unicast**

例：

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

IPv4 アドレス ファミリを指定し、ルータ アドレス ファミリ コンフィギュレーション モードを開始します。

ステップ 5 **mpls ldp sync** [**level** { **1** | **2** }]

例：

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

インターフェイス HundredGigE 0/9/0/0 の IPv4 アドレス ファミリに対して MPLS LDP 同期をイネーブルにします。

ステップ 6 **commit**

IS-IS 過負荷ビット無効化

IS-IS過負荷ビット無効化機能により、ネットワーク管理者は、ラベル スイッチド パス（LSP）内のルータに Intermediate System-to-Intermediate System（IS-IS）の過負荷ビットが設定されているときにパスがディセーブルになることを防止できます。

IS-IS 過負荷ビット無効化機能がアクティブ化されると、過負荷ビットが設定されているすべてのノード（先頭ノード、中間ノード、終端ノードを含む）は無視されます。つまり、それらはラベル スイッチド パス（LSP）で使用できます。

（注） IS-IS 過負荷ビット無効化機能は、ノードがパス計算（PCALC）に含まれていない場合には、過負荷ビットが設定されたノードのデフォルトの動作を変更しません。

IS-IS 過負荷ビット無効化機能は、次のコマンドでアクティブ化されます。

mpls traffic-eng path-selection ignore overload

IS-IS 過負荷ビット無効化機能は、このコマンドの **no** 形式で非アクティブ化されます。

no mpls traffic-eng path-selection ignore overload

IS-IS 過負荷ビット無効化機能が非アクティブ化されると、過負荷ビットが設定されたノードは最終手段のノードとして使用されません。

IS-IS 過負荷ビット無効化の設定

ここでは、IS-IS 過負荷ビット無効化をアクティブにする方法について説明します。

始める前に

IS-IS 過負荷ビット無効化機能は、次の機能をサポートするネットワークでのみ有効です。

- MPLS
- IS-IS

手順

ステップ 1 **configure**

ステップ 2 **mpls traffic-eng path-selection ignore overload**

例：

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

IS-IS 過負荷ビット無効化をアクティブにします。

IS-IS 過負荷ビット無効化の設定：例

次に、IS-IS 過負荷ビット無効化をアクティブにする例を示します。

```
config
 mpls traffic-eng path-selection ignore overload
```

次に、IS-IS 過負荷ビット無効化を非アクティブにする例を示します。

```
config
 no mpls traffic-eng path-selection ignore overload
```

IS-IS の参照

この項では、IS-IS に関する追加の概念情報について説明します。説明する項目は次のとおりです。

- IS-IS 機能の概要 （38 ページ）
- デフォルト ルート （39 ページ）
- ルータの過負荷ビット （39 ページ）
- IS-IS インスタンスの attached ビット （40 ページ）
- ルート タグの IS-IS サポート （40 ページ）
- 特定のインターフェイスでのフラッディングのブロック （40 ページ）
- マルチインスタンス IS-IS （41 ページ）

IS-IS 機能の概要

小規模の IS-IS ネットワークは、一般的にネットワーク内にすべてのルータが含まれる単一のエリアとして構築されます。ネットワークの規模が大きくなるにしたがって、このネットワークは、すべてのエリアに属する、接続されたすべてのレベル 2 ルータのセットから構成されるバックボーン エリア内に再編成され、その後、このネットワークはローカル エリアに接続されます。ローカル エリア内部では、すべてのルータがすべてのシステム ID に到達する方法を認識しています。エリア間では、ルータはバックボーンへの到達方法を認識しており、バックボーン ルータは他のエリアに到達する方法を認識しています。

IS-IS ルーティング プロトコルは、バックボーンのレベル 2 とレベル 1 エリアの構成、および必要とされるエリア間のルーティング情報の移動をサポートします。ルータはレベル 1 隣接を確立して、ローカルエリア内でルーティングを実行します（エリア内ルーティング）。ルータはレベル 2 隣接を確立して、レベル 1 エリア間でルーティングを実行します（エリア間ルーティング）。

各 IS-IS インスタンスは、レベル 1 またはレベル 2 エリアを 1 つだけサポートするか、またはそれぞれのエリアを 1 つずつサポートできます。デフォルトでは、すべての IS-IS インスタンスが自動的にレベル 1 およびレベル 2 ルーティングをサポートします。特定のルーティング インスタンスによって実行されるルーティングのレベルを変更するには、is-type コマンドを使用します。

機能制限

IS-IS の複数のインスタンスが実行されている場合、インターフェイスは 1 インスタンス（プロセス）だけに関連付けることができます。インスタンスは、インターフェイスを共有できません。

デフォルト ルート

デフォルト ルートを IS-IS ルーティング ドメインに強制することができます。IS-IS ルーティング ドメインへのルートの再配布を明確に設定しても、デフォルトではソフトウェアが IS-IS ルーティング ドメインにデフォルト ルートを再配布することはありません。**default-information originate** コマンドを使用すると、IS-IS にデフォルト ルートが生成され、ルート ポリシーで制御できます。ルート ポリシーを使用してデフォルト ルートが通知されるレベルを決定できます。また、ルート ポリシーによって設定できる他のフィルタリング オプションを指定できます。ルート ポリシーを使用することにより、ルータのルーティング テーブル内での他のルートの存在に応じて、デフォルト ルートを条件付きでアドバタイズできます。

ルータの過負荷ビット

過負荷ビットはステート情報の固有ビットであり、ルータの LSP に含まれます。ルータにこのビットが設定されると、このルータがトラフィックの中継に利用できないことがエリア内のルータに通知されます。この機能は次の 4 つの状況で役立ちます。

1. 深刻だが致命的ではないエラーの発生中（メモリ不足など）。
2. プロセスの起動中および再起動中。ルーティング プロトコルが収束するまで過負荷ビットを設定できます。ただし通常の NSF 再起動またはフェールオーバーの最中は使用しません。使用するとルーティング フラップの原因になります。
3. 新しいルータの試験的な導入の最中。導入が検証されるまで過負荷ビットを設定できます。検証後ビットを消去します。
4. ルータのシャットダウン中。ルータのサービスを停止する前に、トポロジからルータを削除するために過負荷ビットを設定できます。

マルチトポロジ動作中の過負荷ビット設定

過負荷ビットは、シングル トポロジの転送に適用されるため、マルチトポロジ操作中に IPv4 および IPv6 に別々に設定およびクリアされる場合があります。このため、過負荷は、ルータ アドレス ファミリ コンフィギュレーション モードで設定されます。IPv4 過負荷ビットが設定されると、エリア内のすべてのルータは、IPv4 の中継トラフィックにこのルータを使用しません。ただし、引き続き IPv6 の中継トラフィックにはこのルータを使用できます。

IS-IS インスタンスの attached ビット

attached ビットは is-type コマンドと **level-1-2** キーワードでルータに設定します。attached ビットはルータが他のエリアに接続されていることを示します（通常はバックボーン経由）。この機能は、ルータがバックボーンへのデフォルトルートとして領域のレベル1ルータから使用できることを意味します。attached ビットは通常、ルータが他のエリアを検出時にレベル2の SPF ルートを計算する間に自動的に設定されます。このビットはルータがバックボーンから切断されると自動的に消去されます。

（注） レベル2インスタンスの接続が失われた場合、レベル1インスタンスの LSP 内の attached ビットによってレベル2インスタンスへのトラフィックの送信が続けられ、トラフィックのドロップを発生させます。

level-1-2 キーワードの機能を表すために複数のプロセスを使用するときにこの動作をシミュレートするには、レベル 1 プロセスの attached ビットを手動で設定します。

ルート タグの IS-IS サポート

ルート タグの IS-IS サポート機能によって、IS-IS ルート プレフィックスとタグを関連付けてアドバタイズする機能が提供されます。また、この機能により、RIB内のルートプレフィックスのインストール順序のプライオリティ付けを、ルートのタグに基づいて行うことができます。ルート タグはまた、ルート ポリシーでルート プレフィックスの照合に使用される可能性があります（たとえば、再配布に特定のルート プレフィックスを選択する場合）。

特定のインターフェイスでのフラッディングのブロック

この手法では、特定のインターフェイスでLSPフラッディングの使用がブロックされますが、残りのインターフェイスではフラッディングに関して通常どおり動作します。この手法は理解しやすく設定も容易ですが、長期的にはメッシュ グループに比べて維持が難しく、エラーが起こりやすくなります。IS-IS で使用するフラッディング トポロジは、制限するのではなく詳細に調整します。トポロジの制限が多すぎると（多くのインターフェイスをブロックしすぎると）障害時にネットワークの信頼性が失われます。トポロジの制限が少なすぎると（ブロックするインターフェイスが少なすぎると）望ましいスケーラビリティが達成できなくなります。

ブロックされていないすべてのインターフェイスでドロップする場合にネットワークの堅牢性を高めるには、インターフェイス コンフィギュレーション モードで csnp-interval コマンドを使用して、ブロックされているポイントツーポイントリンクで定期的にComplete Sequence Number PDU（CSNP）パケットが使用されるようにします。定期的な CSNP によって、ネットワークの同期が可能になります。

最大 LSP ライフタイムおよび更新間隔

デフォルトでは、ルータは定期的な LSP 更新を 15 分ごとに送信します。LSP はデフォルトで 20 分間、データベースに残ります。そのときまでにリフレッシュされない場合、削除されま

す。LSP 更新間隔、または最大 LSP ライフタイムを変更できます。LSP 間隔は、LSP ライフタイムより短くする必要があります。そうしないと、リフレッシュ前に LSP がタイムアウトします。設定された更新間隔がない場合、LSP のタイムアウトを防止するために、必要により LSP 更新間隔がソフトウェアによって調整されます。

メッシュ グループの設定

メッシュ グループ（ルータのインターフェイスのセット）を設定すると、フラッディングを制限できます。特定のメッシュ グループに属するインターフェイスを介して到達可能なすべてのルータには、他のすべてのルータと少なくとも 1 つのリンクがあり、各ルータと緊密に接続されていると見なされます。多数のリンクで障害が発生しても、ネットワークから 1 つまたは複数のルータが切り離されることはありません。

通常のフラッディングでは、新しい LSP が 1 つのインターフェイスで受信されると、そのルータの他のすべてのインターフェイスでフラッディングされます。メッシュ グループでは、メッシュ グループに属する 1 つのインターフェイスで新しい LSP が受信されると、新しい LSP は、そのメッシュ グループに属する他のインターフェイスではフラッディングされません。

マルチインスタンス IS-IS

最大 5 つの IS-IS インスタンスを構成できます。IS-IS プロセスが異なるインターフェイス セット上で実行されている場合には、複数の IS-IS プロセス上で MPLS を実行できます。各インターフェイスは 1 つの IS-IS インスタンスとだけ関連付けられます。ソフトウェアは、設定時に 2 つのインスタンスによるインターフェイスの二重登録を防止します。2 つの MPLS のインスタンスを設定するとエラーになります。

ルーティング情報ベース（RIB）では、各 IS-IS インスタンスは同じルーティング クライアントとして扱われるため、IS-IS インスタンス間でルートを再配布するときには注意が必要です。RIB ではレベル 1 ルートがレベル 2 ルートよりも優先されることが認識されません。このためレベル 1 とレベル 2 のインスタンスを実行する場合には、2 つのインスタンスに異なるアドミニストレーティブ ディスタンスを設定して強制的に優先する必要があります。

ラベル配布プロトコル IGP 自動設定

ラベル配布プロトコル（LDP）内部ゲートウェイ プロトコル（IGP）自動設定は、IGP インスタンスに使用される一連のインターフェイス上で LDP をイネーブルにする手順を簡素化します。LDP IGP 自動設定は、多数のインターフェイス（LDP がコア内の転送に使用される場合など）および複数の IGP インスタンス上で同時に使用できます。

この機能は、デフォルトの VPN ルーティングおよび転送（VRF）インスタンスとして IPv4 アドレス ファミリをサポートします。

LDP IGP 自動設定は、LDP の個々のインターフェイス上で igp auto-config disable コマンド使用して明示的にディセーブルにすることもできます。これにより LDP は、明示的にディセーブルにされたインターフェイスを除くすべての IGP インターフェイスで受信できます。

LDP IGP 自動設定の設定については、『*MPLS configuration guide*』を参照してください。

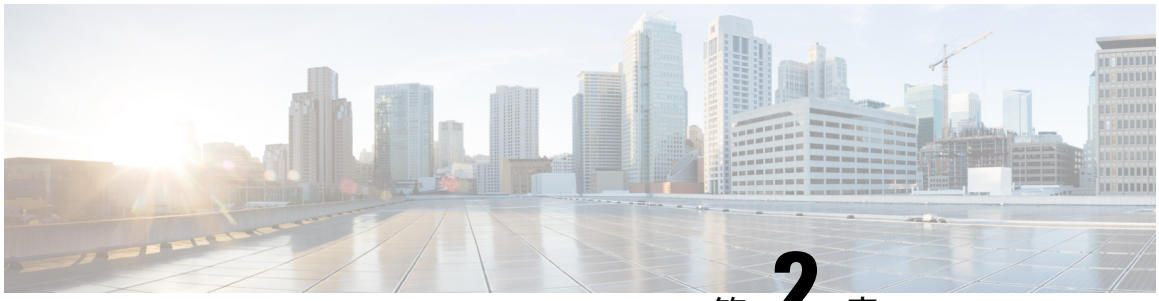
LDP グレースフル リスタートとの MPLS LDP-IGP 同期の互換性

LDP グレースフル リスタートは、LDP セッションが失われた場合にトラフィックを保護します。グレースフル リスタートがイネーブルである LDP セッションに障害が発生した場合でも、グレースフル リスタートで保護されている間、インターフェイス上で MPLS LDP IS-IS 同期が実現されます。MPLS LDP IGP 同期は次の状況で最終的に失われます。

- LDP グレースフル リスタートの再接続タイマーが期限切れになる前に、LDP を再起動できない場合。
- LDP グレースフル リスタートの回復タイマーが期限切れになる前に、保護されたインターフェイス上の LDP セッションを回復できない場合。

IGP ノンストップ フォワーディングとの MPLS LDP-IGP 同期の互換性

IS-IS ノンストップ フォワーディング（NSF）は、IS-IS プロセスの再起動中およびルート プロセッサ（RP）のフェールオーバー中にトラフィックを保護します。LDP IS-IS 同期は、インターフェイス上で LDP グレースフル リスタートもイネーブルの場合のみ IS-IS NSF とともにサポートされます。IS-IS NSF がイネーブルでない場合、LDP の同期状態は再起動およびフェールオーバーの際に維持されません。



第 2 章

OSPF の実装

Open Shortest Path First（OSPF）は、Internet Engineering Task Force（IETF）の OSPF ワーキング グループによって開発された内部ゲートウェイ プロトコル（IGP）です。OSPF は特に IP ネットワーク向けに設計されており、IP サブネット化、および外部から取得したルーティング情報のタギングをサポートしています。また、OSPF では、パケットの送受信時のパケット認証も可能になります。

OSPF Version 3（OSPFv3）は OSPF Version 2 を拡張し、IPv6 ルーティング プレフィックスのサポートを提供します。

このモジュールでは、ソフトウェアで OSPF の両方のバージョンを実装するために必要な概念と作業について説明します。特に記載のないかぎり、用語「OSPF」は両方のバージョンのルーティング プロトコルを意味します。

（注）

1. VPNv4、VPNv6 および VPN ルーティング/転送（VRF）のアドレス ファミリは、今後のリリースでサポートされる予定です。
2. GTSM TTL セキュリティはサポートされていません。

- OSPF の実装の前提条件 （44 ページ）
- OSPF のイネーブル化 （44 ページ）
- OSPF の設定と動作の確認 （47 ページ）
- スタブ エリア （49 ページ）
- OSPF のネイバーおよび隣接関係 （52 ページ）
- 認証方法 （56 ページ）
- OSPF に同じ LSA が生成される頻度または受け入れられる頻度の制御 （60 ページ）
- OSPF の仮想リンクおよび中継エリア （61 ページ）
- OSPF ABR でのサブネットワーク LSA の集約 （67 ページ）
- OSPF のルート再配布 （69 ページ）
- OSPF Version 2 のノンストップ フォワーディング （72 ページ）
- OSPF Shortest Path First スロットリング （74 ページ）
- OSPFv3 のグレースフル リスタート （77 ページ）

- OSPF Version 2 のウォーム スタンバイとノンストップ ルーティング （80 ページ）
- OSPF Version 3 のウォーム スタンバイとノンストップ ルーティング （81 ページ）
- OSPFv2OSPF SPF プレフィックスのプライオリティ設定 （82 ページ）
- プロバイダー エッジからカスタマー エッジ（PE-CE）プロトコルとしての OSPF の設定 （86 ページ）
- 複数の OSPF インスタンスの作成（OSPF プロセスおよび VRF） （88 ページ）
- OSPF のラベル配布プロトコル IGP 自動設定 （90 ページ）
- OSPF 認証のメッセージ ダイジェスト管理 （93 ページ）
- OSPF の GTSM TTL セキュリティ メカニズム （96 ページ）
- OSPF の参照 （99 ページ）

OSPF の実装の前提条件

以下は、OSPF を実装するための前提条件です。

- OSPFv3 の設定作業では、IPv6 のアドレッシングと基本概念について精通していることを前提としています。IPv6 ルーティングとアドレッシングについては、『*Cisco IP Addresses and Services Configuration Guide IP Addresses and Services Configuration Guide for Cisco NCS 560 Series Routers*』の「*Implementing Network Stack IPv4 and IPv6*」を参照してください。
- インターフェイスで OSPFv3 をイネーブルにする前に、次の手順を実行する必要があります。
  - ご使用の IPv6 ネットワークに対する OSPF ネットワーク戦略と計画を完成させます。たとえば、複数のエリアが必要かどうかを決定します。
  - インターフェイスで IPv6 をイネーブルにします。
- 認証（IP セキュリティ）の設定はオプションの作業です。認証を設定する場合、プレーンテキスト認証と Message Digest 5（MD5）認証のどちらを設定するかについて、また、認証をエリア全体に適用するか特定のインターフェイスに適用するかについて最初に決定する必要があります。

OSPF のイネーブル化

このタスクでは、1 つのルータ ID で OSPF プロセスをイネーブルにするルータで、最小の OSPF コンフィギュレーションを実行し、バックボーンまたはバックボーン以外のエリアを設定し、OSPF を実行する 1 つ以上のインターフェイスを割り当てるする方法を説明します。

始める前に

IP アドレスを設定する前に OSPF を設定することはできますが、IP アドレスが設定されるまで、OSPF はルーティングされません。

手順

ステップ 1 **configure**

ステップ 2 次のいずれかを実行します。

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

例：

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

または

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

指定したルーティング プロセスに OSPF ルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

または

指定したルーティング プロセスに OSPFv3 ルーティングをイネーブルにし、router ospfv3 コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **router-id** { *router-id* }

例：

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

OSPF プロセスのルータ ID を設定します。

（注） 固定 IP アドレスをルータ ID として使用することを推奨します。

ステップ 4 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始し、OSPF プロセスのエリアを設定します。

- バックボーン エリアには 0 のエリア ID があります。
- バックボーン以外のエリアにはゼロではないエリア ID があります。
- *area-id* 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。IPv4 アドレス形式を使用することを推奨します。

ステップ 5 **interface** *type interface-path-id*

例：

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

インターフェイス コンフィギュレーション モードを開始して、ステップ 4 で設定したエリアのインターフェイスを 1 つ以上関連付けます。

ステップ 6 OSPF を使用する各インターフェイスでステップ 5 を繰り返します。

—

ステップ 7 **log adjacency changes** [**detail**] [**enable** | **disable**]

例：

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

（任意）ネイバー変更の通知を要求します。

- デフォルトでは、この機能はイネーブルです。
- ネイバー変更によって生成されたメッセージは通知と見なされます。このメッセージは logging console コマンドで重大度レベル 5 に分類されます。logging console コマンドではどの重大度レベルのメッセージをコンソールに送信するかを制御します。デフォルトでは、すべての重大度レベルのメッセージが送信されます。

ステップ 8 **commit**

OSPF のイネーブル化：例

OSPF エリアは明示的に設定する必要があり、エリア コンフィギュレーション モードで設定されたインターフェイスは、そのエリアに明示的にバインドされています。この例では、インターフェイス 10.1.2.0/24 がエリア 0 に、インターフェイス 10.1.3.0/24 がエリア 1 にバインドされています。

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

OSPF の設定と動作の確認

このタスクでは、OSPF の設定および操作を確認する方法について説明します。

手順

ステップ 1 **show** { **ospf** | **ospfv3** } [*process-name*]

例：

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

（任意）OSPF ルーティング プロセスに関する一般情報を表示します。

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

例：

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

（任意）ABR および ASBR への内部 OSPF ルーティング テーブル エントリを表示します。

ステップ 3 **show** { **ospf** | **ospfv3** } [*process-name*] **database**

例：

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

（任意）特定のルータの OSPF データベースに関係する情報の一覧を表示します。

- このコマンドは、さまざまな形式で、異なる OSPF LSA に関する情報を提供します。

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

例：

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

（任意）インターフェイス上でのフラッディングを待機している OSPF LSA のリストを表示します。

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

例：

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

例：

```
RP/0/RP0/CPU0:router# show ospf interface
Interfaces for OSPF test
TenGigE0/0/0/7 is up, line protocol is up
  Internet Address 7.7.7.3/16, Area 1
```

```
Label stack Primary label 0 Backup label 0 SRTE label 0
Process ID test, Router ID 7.7.7.3, Network Type BROADCAST, Cost: 1 (RSVP-TE)
Transmit Delay is 1 sec, State DR, Priority 1, MTU 1500, MaxPktSz 1500
Forward reference No, Unnumbered no,  Bandwidth 10000000
Designated Router (ID) 7.7.7.3, Interface address 7.7.7.3
Backup Designated router (ID) 7.7.7.2, Interface address 7.7.7.2
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
  Hello due in 00:00:01:801
Index 1/1, flood queue length 0
Next 0(0)/0(0)
Last flood scan length is 2, maximum is 2
Last flood scan time is 0 msec, maximum is 0 msec
LS Ack List: current length 0, high water mark 1
Neighbor Count is 1, Adjacent neighbor count is 1
  Adjacent with neighbor 7.7.7.2  (Backup Designated Router)
Suppress hello for 0 neighbor(s)
Keychain-based authentication enabled
  Key id used is 1
  Cryptographic algorithm HMAC-SHA-256
Multi-area interface Count is 0
```

（任意）OSPF インターフェイス情報を表示します。

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

例：

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

（任意）個々のインターフェイスに基づいた OSPF ネイバー情報を表示します。

ステップ 7 **clear** { **ospf** | **ospfv3** }[*process-name*] **process**

例：

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

（任意）OSPF ルータ プロセスを停止および再起動せずにリセットします。

ステップ 8 **clear**{**ospf**|**ospfv3**[*process-name*] **redistribution**

例：

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

OSPF ルート再配布をクリアします。

ステップ 9 **clear**{**ospf**|**ospfv3**[*process-name*] **routes**

例：

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

OSPF ルート テーブルをクリアします。

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

例：

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

OSPF ルート テーブルをクリアします。

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

例：

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

（任意）ネイバー状態遷移の OSPF 統計情報をクリアします。

スタブ エリア

スタブ エリアは、ルート アドバタイズメントや、エリアの外部にあるネットワークの詳細情報を受け入れないエリアです。スタブエリアには、通常、エリアと他の自律システムとのインターフェイスになるルータが 1 つだけがあります。スタブ ABR は、外部の宛先への単一のデフォルト ルートをスタブ エリアにアドバタイズします。スタブ エリア内のルータはエリア外の宛先および自律システムに対してこのルートを使用します。この関係は、エリアにフラッディングされた外部 LSA を格納するためにも使用される LSA データベースのスペースを節約します。

Not-So-Stubby Area

Not-So-Stubby Area（NSSA）はスタブ エリアに似ています。NSSA はコアからエリアへとタイプ 5 の外部 LSA をフラッディングしませんが、限定的に自律システム外部ルートをエリア内にインポートできます。

NSSA は、再配布によって、タイプ 7 の自律システムの外部ルートを NSSA エリア内部にインポートできます。これらのタイプ 7 の LSA は、NSSA の ABR によってタイプ 5 の LSA に変換され、ルーティング ドメイン全体にフラッディングされます。変換中は集約とフィルタリングがサポートされます。

異なるルーティング プロトコルを使用するリモート サイトに OSPF を使用する中央サイトを接続する必要があるネットワーク管理者であれば、管理を簡素化するために NSSA 使用します。

スタブエリアにはリモート サイトのルートが再配布されないため、NSSA が実装される前は、企業サイトの境界ルータとリモート ルータ間の接続に OSPF スタブ エリアを利用できず、2 つのルーティング プロトコルを維持する必要がありました。RIP のようなシンプルなプロトコルを実行して再配布を処理する方法が一般的でした。NSSA が実装されたことで、企業ルータとリモート ルータ間のエリアを NSSA として定義することにより、NSSA で OSPF を拡張してリモート接続をカバーできます。エリア 0 を NSSA にすることはできません。

スタブ エリア タイプおよび Not-So-Stubby Area タイプの設定

このタスクでは、OSPF のスタブ エリアおよび NSSA を設定する方法を説明します。

手順

ステップ 1 **configure**

ステップ 2 次のいずれかを実行します。

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

例：

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

または

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

指定したルーティング プロセスに OSPF ルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

または

指定したルーティング プロセスに OSPFv3 ルーティングをイネーブルにし、router ospfv3 コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **router-id** { *router-id* }

例：

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

OSPF プロセスのルータ ID を設定します。

（注） 固定 IP アドレスをルータ ID として使用することを推奨します。

ステップ 4 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始し、OSPF プロセスのバックボーン以外のエリアを設定します。

- *area-id* 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。IPv4 アドレス形式を使用することを推奨します。

ステップ 5 次のいずれかを実行します。

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

例：

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

または

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

非バックボーン エリアをスタブ エリアとして定義します。

- スタブ エリアに送信される LSA の数をさらに減らすために **no-summary** キーワードを指定します。このキーワードにより、ABR がサマリー リンクステート アドバタイズメント（タイプ 3）をスタブ エリアに送信しないようにします。

または

エリアを NSSA として定義します。

ステップ 6 次のいずれかを実行します。

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

例：

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

または

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

（任意）スタブ エリア、および NSSA エリアに設定されたオプションをオフにします。

- ステップ 5 でオプションのキーワード（ **no-summary** 、 **no-redistribution** 、 **default-information-originate** 、および **no-summary** ）を使用してスタブ エリアおよび NSSA エリアを設定した場合、コマンドの **no** 形式を使用するのではなく、**stub** および **nssa** コマンドをこれらのキーワードなしで再度発行する必要があります。
- たとえば、コマンドの**no nssa default-information-originate** 形式は、NSSA エリアを通常のエリアに変更し、そのエリアの既存の隣接関係を意図せずダウンさせます。

ステップ 7 **default-cost** *cost*

例：

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

（任意）スタブ エリアまたは NSSA に送信されるデフォルト サマリー ルートのコストを指定します。

- このコマンドはNSSAにアタッチされているABRでのみ使用します。エリア内の他のルータには使用しないでください。
- デフォルトのコストは 1 です。

ステップ 8 **commit**

ステップ 9 スタブ エリアまたは NSSA にある他のすべてのルータでこのタスクを繰り返します。

—

スタブ エリアの設定：例

エリア 1 がスタブ エリアとして設定される例を次に示します。

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

OSPF のネイバーおよび隣接関係

セグメントを共有するルータ（2 つのインターフェイス間のレイヤ 2 リンク）は、そのセグメント上でネイバー同士となります。OSPF では Hello プロトコルをネイバー探索およびキープアライブ メカニズムとして使用します。Helloプロトコルでは定期的にhelloパケットを各インターフェイスで送受信します。helloパケットは、インターフェイス上のすべての既知の OSPF ネイバーをリストします。ルータがネイバーの Hello パケット内に自身がリストされていることを認識すると、それらのルータはネイバー同士となります。2 つのルータがネイバーになると、データベースの交換や同期化を行うことができるようになります。これにより、隣接が作成されます。ブロードキャストおよび NBMA ネットワークのすべての隣接ルータに隣接があります。

非ブロードキャスト ネットワークのネイバーの設定

このタスクでは、非ブロードキャスト ネットワークにネイバーを設定する方法を説明します。このタスクはオプションです。

始める前に

NBMAネットワークをブロードキャストまたは非ブロードキャストとして構成する場合は、各ルータから各ルータあるいはフルメッシュのネットワークにまで仮想回線があると想定されます。

手順

ステップ 1 **configure**

ステップ 2 次のいずれかを実行します。

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

例：

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

または

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

指定したルーティング プロセスに OSPF ルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

または

指定したルーティング プロセスに OSPFv3 ルーティングをイネーブルにし、router ospfv3 コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **router-id** { *router-id* }

例：

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

OSPF プロセスのルータ ID を設定します。

（注） 固定 IP アドレスをルータ ID として使用することを推奨します。

ステップ 4 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始し、OSPF プロセスのエリアを設定します。

- この例ではバックボーン エリアを設定します。
- *area-id* 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。IPv4 アドレス形式を使用することを推奨します。

ステップ 5 **network** { **broadcast** | **non-broadcast** }

例：

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

OSPF ネットワーク タイプをそのメディアのデフォルト以外のタイプに設定します。

- この例では、ネットワーク タイプを NBMA に設定します。

ステップ 6 **dead-interval** *seconds*

例：

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

（任意）ネイバーのダウンを宣言する前に、ネイバーからの hello パケットを待機する時間を設定します。

ステップ 7 **hello-interval** *seconds*

例：

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

（任意）OSPF がインターフェイスで送信する hello パケットの間隔を指定します。

ステップ 8 **interface** *type interface-path-id*

例：

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

インターフェイス コンフィギュレーション モードを開始して、ステップ 4 で設定したエリアのインターフェイスを 1 つ以上関連付けます。

- この例では、値がインターフェイス レベルで設定されていないため、インターフェイスはブロードキャスト ネットワーク タイプおよび hello および dead 間隔をそのエリアから継承します。

ステップ 9 次のいずれかを実行します。

- **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**]]

例：

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

または

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

ブロードキャスト ネットワーク以外と相互接続する OSPF ネイバーの IPv4 アドレスを設定します。

または

OSPFv3 ネイバーのリンクローカル IPv6 アドレスを設定します。

- *ipv6-link-local-address* 引数は、RFC 2373 に記載されている形式である必要があります。このアドレスは 16 ビット値を使用する 16 進数をコロンで区切って指定します。
- **priority** キーワードでは、このネイバーが DR または BDR になる資格があることをルータに通知します。priority 値は隣接ルータの実際のプライオリティ設定と一致する必要があります。ネイバー プライオリティのデフォルト値はゼロです。
- 特定のコストが設定されていないネイバーは、cost コマンドに基づいてインターフェイスのコストを想定します。
- **database-filter** キーワードでは OSPF ネイバーへの発信 LSA をフィルタリングします。**all** キーワードを指定すると、着信および発信 LSA はフィルタリングされます。

ステップ 10 インターフェイスのすべてのネイバーでステップ 9 を繰り返します。

—

ステップ 11 exit

例：

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

エリア コンフィギュレーション モードを開始します。

ステップ 12 **interface** *type interface-path-id*

例：

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

インターフェイス コンフィギュレーション モードを開始して、ステップ 4 で設定したエリアのインターフェイスを 1 つ以上関連付けます。

- この例では、値がインターフェイス レベルで設定されていないため、インターフェイスはブロードキャスト ネットワーク タイプおよび hello および dead 間隔をそのエリアから継承します。

ステップ 13 次のいずれかを実行します。

- **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**]]

例：

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

または

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

ブロードキャスト ネットワーク以外と相互接続する OSPF ネイバーの IPv4 アドレスを設定します。

または

OSPFv3 ネイバーのリンクローカル IPv6 アドレスを設定します。

- *ipv6-link-local-address* 引数は、RFC 2373 に記載されている形式である必要があります。このアドレスは 16 ビット値を使用する 16 進数をコロンで区切って指定します。
- **priority** キーワードでは、このネイバーが DR または BDR になる資格があることをルータに通知します。priority 値は隣接ルータの実際のプライオリティ設定と一致する必要があります。ネイバー プライオリティのデフォルト値はゼロです。
- 特定のコストが設定されていないネイバーは、cost コマンドに基づいてインターフェイスのコストを想定します。
- **database-filter** キーワードでは OSPF ネイバーへの発信 LSA をフィルタリングします。**all** キーワードを指定すると、着信および発信 LSA はフィルタリングされます。フィルタリングによりルーティング トポロジが 2 つのネイバー間でまったく異なるように見え、「black-holing」化やルーティング ループを引き起こしたりすることがあるため、十分注意して使用してください。

ステップ 14 インターフェイスのすべてのネイバーでステップ 13 を繰り返します。

—

ステップ 15 **commit**

認証方法

プロセスまたはエリアの全体に対して、またはインターフェイスまたは仮想リンク上に認証を指定できます。インターフェイスまたは仮想リンクには 1 種類の認証だけを設定でき、両方とも設定することはできません。エリアまたはプロセス用に設定されたインターフェイスまたは仮想リンクのオーバーライド認証用に設定された認証。

エリアのすべてのインターフェイスで同じ認証タイプを使用する場合、エリア コンフィギュレーション サブモードで **authentication** コマンドを使用すると（また、エリア全体で MD5 認証を使用する場合は **message-digest** キーワードを指定すると）、より少ないコマンドを設定できます。この方法を使用すると、各インターフェイスに認証を指定するときに必要なコマンドよりも少ないコマンドで設定できます。

OSPF Version 2 の異なる階層レベルでの認証の設定

このタスクでは、OSPF ルータ プロセスに MD5（セキュア）認証を設定する方法について説明します。プレーン テキスト認証を 1 エリアに設定し、次にクリア テキスト（null）認証を 1 インターフェイスに適用します。

（注） インターフェイス レベルで設定された認証は、エリア レベルおよびルータ プロセス レベルで設定された認証を上書きします。インターフェイスに特別に設定された認証がない場合、そのインターフェイスは認証パラメータ値をより高い階層レベルから継承します。

始める前に

認証を設定する場合、プレーン テキスト認証または MD5 認証のどちらを設定するかをはじめに決定する必要があります。また、認証の適用対象がプロセス内のすべてのインターフェイスか、全エリアか、特定のインターフェイスかを決定する必要があります。ネットワークに特定のメソッドを使用する場合、それぞれの種類の認証に関する情報については、OSPF の階層 CLI および CLI 継承 （101 ページ）を参照してください。

手順

ステップ 1 **configure**

ステップ 2 **router ospf** *process-name*

例：

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

指定したルーティング プロセスに OSPF ルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **router-id** { *router-id* }

例：

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

OSPF プロセスのルータ ID を設定します。

ステップ 4 **authentication** [**message-digest** | **null**]

例：

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

OSPF プロセスに対して MD5 認証が有効になります。

- エリアやインターフェイスなどのより低い階層レベルによって変更されないかぎり、この認証タイプはルータ プロセス全体に適用されます。

ステップ 5 **message-digest-key** *key-id* **md5** { *key* | **clear** *key* | **encrypted** *key* | **LINE**}

例：

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

OSPF プロセスに対して MD5 認証キーを指定します。

- 隣接ルータが、同じキー ID を保持する必要があります。

ステップ 6 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始して、OSPF プロセスのバックボーン エリアを設定します。

ステップ 7 **interface** *type interface-path-id*

例：

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

インターフェイス コンフィギュレーション モードを開始して、1 つ以上のインターフェイスをバックボーン エリアに関連付けます。

- すべてのインターフェイスは、OSPF プロセスの指定された認証パラメータ値を継承します（ステップ 4、ステップ 5、ステップ6）。

ステップ 8 同じ認証を使用して通信する必要があるインターフェイスごとにステップ 7 を繰り返します。

—

ステップ 9 **exit**

例：

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

エリア OSPF コンフィギュレーション モードを開始します。

ステップ 10 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始し、OSPF プロセスの非バックボーンのエリア 1 を設定します。

- *area-id* 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。IPv4 アドレス形式を使用することを推奨します。

ステップ 11 **authentication** [**message-digest** | **null**]

例：

```
RP/0/RP0/cpu 0: router(config-ospf-ar)# authentication
```

セキュリティのないタイプ 1（プレーン テキスト）認証をイネーブルにします。

- 例では、プレーンテキスト認証を（キーワードを指定しないことによって）指定します。インターフェイス コンフィギュレーション モードで **authentication-key** コマンドを使用し、このプレーン テキスト パスワードを指定します。

ステップ 12 **interface** *type interface-path-id*

例：

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

インターフェイス コンフィギュレーション モードを開始して、ステップ 7 で指定したバックボーン以外のエリア 1 に 1 つ以上のインターフェイスを関連付けます。

- 設定されているすべてのインターフェイスがエリア 1 に対して設定されている認証パラメータ値を継承します。

ステップ 13 同じ認証を使用して通信する必要があるインターフェイスごとにステップ 12 を繰り返します。

—

ステップ 14 **interface** *type interface-path-id*

例：

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

インターフェイス コンフィギュレーション モードを開始し、1 つ以上のインターフェイスを異なる認証タイプに関連付けます。

ステップ 15 **authentication** [**message-digest** | **null**]

例：

```
RP/0/RP0/cpu 0: router(config-ospf-ar-if)# authentication null
```

TenGigE 0/0/0/0 に認証なしを指定し、エリア 1 に指定されたプレーン テキスト認証を上書きします。

- デフォルトでは、同じエリアで設定されるすべてのインターフェイスは、エリアと同じ認証パラメータ値を継承します。

ステップ 16 **commit**

OSPFに同じLSAが生成される頻度または受け入れられる頻度の制御

このタスクでは、非常に短い間隔で多数の LSA がフラッディングされる必要がある場合に、ルーティングテーブルのOSPFルートのコンバージェンス時間を調整する方法を説明します。

手順

ステップ 1 **configure**

ステップ 2 次のいずれかを実行します。

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

例：

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

または

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

指定したルーティングプロセスにOSPFルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

または

指定したルーティング プロセスに OSPFv3 ルーティングをイネーブルにし、router ospfv3 コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **router-id** { *router-id* }

例：

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

OSPF プロセスのルータ ID を設定します。

（注） 固定 IP アドレスをルータ ID として使用することを推奨します。

ステップ 4 ステップ 5 またはステップ 6、または両方のステップを実行して、同じ LSA が送受信される間隔を制御します。

—

ステップ 5 **timers lsa refresh** *seconds*

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# timers lsa refresh 1800
```

自動送信 LSA をリフレッシュする頻度を秒単位で設定します。

- OSPF および OSPFv3 の両方で、デフォルトは 1800 秒です。

ステップ 6 **timers lsa min-arrival** *seconds*

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# timers lsa min-arrival 2
```

フラッディング中に特定の OSPF Version 2 LSA の新しいプロセスが受け入れられる頻度を制限します。

- デフォルト値は 1 秒です。

ステップ 7 **timers lsa group-pacing** *seconds*

例：

```
RP/0/
/CPU0:router(config-ospf)# timers lsa group-pacing 1000
```

OSPF リンクステート LSA がフラッディングのグループに収集される間隔を変更します。

- デフォルトは 240 秒です。

ステップ 8 **commit**

OSPF の仮想リンクおよび中継エリア

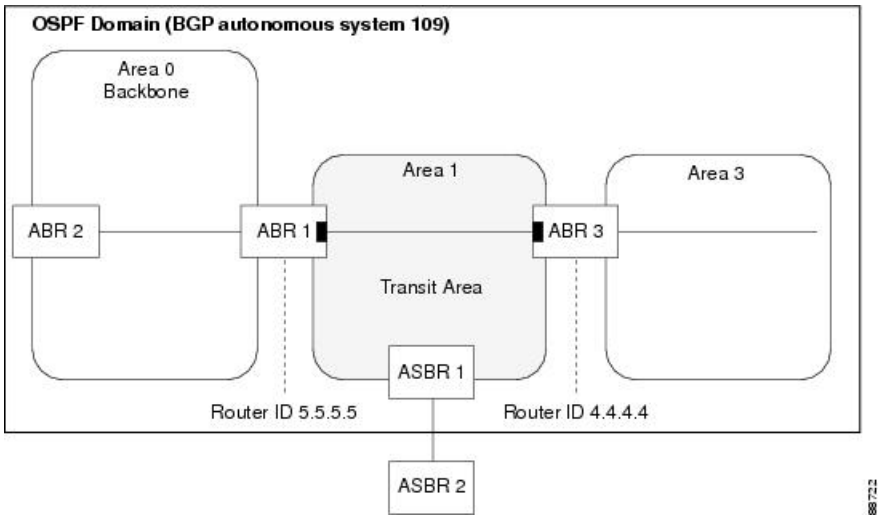
OSPF では、すべてのエリアからのルーティング情報は、ABR によって最初にバックボーンエリアに集約されます。次に、同じ ABR は受信したその情報をアタッチされているエリアに伝播します。このような階層型のルーティング情報の配信では、すべてのエリアがバックボーンエリア（エリア 0）に接続する必要があります。エリアを定義する必要がある場合もありますが、エリア 0 には物理的に接続することはできません。そのような場合の例として、会社で OSPF エリアが含まれる新しい取得を行う場合やエリア 0 自体がパーティション化されている場合が挙げられます。

エリアをエリア 0 に接続できない場合、そのエリアとエリア 0 の間で仮想リンクを設定する必要があります。仮想リンクの 2 つのエンドポイントは ABR であり、仮想リンクは両方のルータで設定する必要があります。2 つのルータが属する、バックボーン以外の共通エリアは中継エリアと呼ばれます。仮想リンクは、他の仮想エンドポイント（他の ABR）の中継エリアとルータ ID を指定します。

仮想リンクはスタブ エリアまたは NSSA から設定することはできません。

図 *1:* エリア *0* への仮想リンク

この図はエリア 3 からエリア 0 への仮想リンクを示します。



仮想リンクの作成

このタスクでは、仮想リンクをバックボーン（エリア 0）に作成して MD5 認証を適用する方法について説明します。説明されている手順は、仮想リンクの各端にある両方の ABR で実行する必要があります。

（注） 明示的にエリアパラメータ値を設定したら、インターフェイスの値を上書きして明示的に設定しないかぎり、その値はそのエリアにバインドされているすべてのインターフェイスに継承されます。

始める前に

MD5 認証が設定された仮想リンクをエリア 0 に作成するには、次の前提条件を満たす必要があります。

- ローカル ルータを設定するリンクの反対の隣接ルータのルータ ID が必要です。ルータ ID を取得するためにリモート ルータで show ospf コマンドまたは show ospfv3 コマンドを実行できます。
- 仮想リンクが正常に機能するには、仮想リンクの各端に固定ルータ ID が必要です。ルータ ID は変更されないようにします。デフォルトでルータ ID を割り当てると、変更される可能性があります. したがって、仮想リンクを設定する前に、次のいずれかのタスクを実行することをお勧めします。
  - ルータ ID を設定するには、router-id コマンドを使用します。この方法を推奨します。

- ルータが安定したルータ ID を持つために、ループバック インターフェイスを設定します。

- OSPF Version 2 の仮想リンクを設定する前に、プレーン テキスト認証、MD5 認証、認証なし（デフォルト）のうち、どの認証を設定するかを決定する必要があります。認証に関連する追加のタスクを実行する必要があるかどうかに応じて決定します。

手順

ステップ 1 次のいずれかを実行します。

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

例：

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

または

```
RP/0//CPU0:router# show ospfv3
```

（任意）OSPF ルーティング プロセスに関する一般情報を表示します。

- 出力にはローカル ルータのルータ ID が表示されます。このルータ ID はリンクのもう一端を設定するために必要です。

ステップ 2 **configure**

ステップ 3 次のいずれかを実行します。

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

例：

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

または

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

指定したルーティング プロセスに OSPF ルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

または

指定したルーティング プロセスに OSPFv3 ルーティングをイネーブルにし、router ospfv3 コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 4 **router-id** { *router-id* }

例：

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

OSPF プロセスのルータ ID を設定します。

（注） 固定 IPv4 アドレスをルータ ID として使用することを推奨します。

ステップ 5 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始し、OSPF プロセスのバックボーン以外のエリアを設定します。

- *area-id* 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。IPv4 アドレス形式を使用することを推奨します。

ステップ 6 **virtual-link** *router-id*

例：

```
RRP/0//CPU0:router(config-ospf-ar)# virtual-link 10.3.4.5
```

OSPF 仮想リンクを定義します。

- を参照してください。

ステップ 7 **authentication message-digest**

例：

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

この仮想リンクに対して MD5 認証を選択します。

ステップ 8 **message-digest-key** *key-id* **md5** { *key* | **clear** *key* | **encrypted** *key* }

例：

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

OSPF 仮想リンクを定義します。

- 仮想リンクを理解するには、を参照してください。
- *key-id* 引数は、1 ～ 255 の範囲の数です。 *key* 引数は最大 16 文字の英数字です。仮想リンクの両端のルータには同じキー ID と、OSPF トラフィックをルーティングできるキーが必要です。
- **authentication-key** *key* コマンドは、OSPFv3 ではサポートされていません。

• キーが暗号化されたら、その暗号化を保持する必要があります。

ステップ 9 仮想リンクの反対側にある ABR でこのタスクのすべての手順を繰り返します。このルータで仮想リンクに指定する同じキー ID およびキーを指定します。

—

ステップ 10 **commit**

ステップ 11 次のいずれかを実行します。

- **show ospf** [*process-name*] [*area-id*] **virtual-links**
- **show ospfv3** [*process-name*] **virtual-links**

例：

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

または

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

（任意）OSPF 仮想リンクのパラメータと現在の状態を表示します。

仮想リンクの作成：例

ABR 1 の設定

ABR 2 の設定

次の例では、show ospfv3 virtual links コマンドで、OSPFv3 ネイバーへの OSPF_VL0 仮想リンクが起動しており、仮想リンク インターフェイスの ID が 2 であり、仮想リンク エンドポイントの IPv6 アドレスが 2003:3000::1 であることを検証します。

```
  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 0/11/0/0 Cost of using 2
  Transmit Delay is 5 sec,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:02
    Adjacency State FULL (Hello suppressed)
    Index 0/2/3, retransmission queue length 0, number of retransmission 1
    First 0(0)/0(0)/0(0) Next 0(0)/0(0)/0(0)
    Last retransmission scan length is 1, maximum is 1
    Last retransmission scan time is 0 msec, maximum is 0 msec

Check for lines:
Virtual Link OSPF_VL0 to router 10.0.0.3 is up
```

```
      Adjacency State FULL (Hello suppressed)

  State is up and Adjacency State is FULL
```

この例では、エリア 0 と 1 および仮想リンク 10.0.0.217 と 10.0.0.212 で構成される OSPFv3 トポロジのエリア 1 からバックボーンを接続するように仮想リンクを設定する方法を説明します。

```
router ospfv3 1
 router-id 10.0.0.217
 area 0
  interface TenGigE 0/11/0/1
 area 1
  virtual-link 10.0.0.212
  interface TenGigE 0/11/0/0


router ospfv3 1
 router-id 10.0.0.212
 area 0
  interface TenGigE 0/11/0/1
 area 1
  virtual-link 10.0.0.217
  interface TenGigE 0/11/0/0
```

この例では、ルータ ABR1 のすべてのインターフェイスは MD5 認証を使用します。

```
router ospf ABR1
 router-id 10.10.10.10
 authentication message-digest
 message-digest-key 100 md5 0 cisco
 area 0
  interface TenGigE 0/11/0/1
  interface TenGigE 0/11/0/0
 area 1
  interface TenGigE 0/11/0/0
  virtual-link 10.10.5.5
!
!
```

この例では、ルータ ABR3 のエリア 1 インターフェイスだけが MD5 認証を使用します。

```
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 0/11/0/1
  virtual-link 10.10.10.10
 area 3
  interface Loopback 0
  interface TenGigE 0/11/0/0
!
```

OSPF ABR でのサブネットワーク LSA の集約

IPアドレスをインターフェイスに割り当てたときに複数のサブネットワークを設定した場合、すべてのサブネットワークが含まれ、ローカルエリアが別のエリアにアドバタイズする1つのLSA にソフトウェアを集約することができます。このようにソフトウェアを集約すると LSA の数を減らすことができるため、ネットワークリソースを節約できます。この集約はエリア間ルート集約と呼ばれます。これは自律システム内のルートに適用されます。再配布によって OSPF に挿入された外部ルートには適用されません。

このタスクでは、一緒にアドバタイズされる範囲に収まるすべてのサブネットワークを指定することによって、サブネットワークを 1 つの LSA に集約するように OSPF を設定します。このタスクは 1 つの ABR でのみ実行します。

手順

ステップ 1 **configure**

ステップ 2 次のいずれかを実行します。

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

例：

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

または

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

指定したルーティング プロセスにOSPFルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

または

指定したルーティング プロセスに OSPFv3 ルーティングをイネーブルにし、router ospfv3 コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **router-id** { *router-id* }

例：

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

OSPF プロセスのルータ ID を設定します。

（注） 固定 IPv4 アドレスをルータ ID として使用することを推奨します。

ステップ 4 **area** *area-id*

例：

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

エリアコンフィギュレーションモードを開始し、OSPFプロセスのバックボーン以外のエリアを設定します。

- *area-id* 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。IPv4 アドレス形式を使用することを推奨します。

ステップ 5 次のいずれかを実行します。

- **range** *ip-address mask* [**advertise** | **not-advertise**]
- **range** *ipv6-prefix / prefix-length* [**advertise** | **not-advertise**]

例：

```
RP/0/RP0/cpu 0: router(config-ospf-ar)# range 192.168.0.0 255.255.0.0 advertise
```

または

```
RP/0/RP0/cpu 0: router(config-ospf-ar)# range 4004:f000::/32 advertise
```

エリア境界で OSPF ルートを統合および集約します。

- **advertise** キーワードにより、タイプ 3 サマリー LSA のソフトウェアがサブネットワークのアドレス範囲をアドバタイズします。
- **not-advertise** キーワードによって、ソフトウェアがタイプ 3 サマリー LSA に制限され、この範囲内のサブネットワークは他のエリアには見えません。
- 最初の例では、ネットワーク 192.168.0.0 のすべてのサブネットワークが ABR によって集約され、バックボーン外のエリアにアドバタイズされます。
- 2 番目の例では、複数の IPv4 インターフェイスが 192.*x.x* のネットワークに対応しています。

ステップ 6 **interface** *type interface-path-id*

例：

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

インターフェイスコンフィギュレーションモードを開始して、1つ以上のインターフェイスをエリアに関連付けます。

ステップ 7 **commit**

例

次の例では、エリア 1 からバックボーンに集約されたプレフィックス範囲 2300::/16 を示します。

```
router ospfv3 1
 router-id 192.168.0.217
 area 0
  interface TenGigE 0/11/0/0
 area 1
  range 2300::/16
  interface TenGigE 0/11/0/0
```

OSPF のルート再配布

再配布により、異なるルーティングプロトコルを使用してルーティング情報を交換できます。この手法を使用すると、複数のルーティング プロトコルに接続を広げることができます。redistribute コマンドでは、OSPF からの再配布ではなく、OSPF プロセスへの再配布が制御されることに注意することが重要です。

OSPF へのルートの再配布

このタスクでは、IGP（別の OSPF プロセスでも可）から OSPF にルートを再配布します。

手順

ステップ 1 **configure**

ステップ 2 次のいずれかを実行します。

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

例：

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

または

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

指定したルーティングプロセスにOSPFルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

または

指定したルーティング プロセスに OSPFv3 ルーティングをイネーブルにし、router ospfv3 コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **router-id** { *router-id* }

例：

```
RRP/0/RP0/cpu 0: router(config-ospf)# router-id 192.168.4.3
```

OSPF プロセスのルータ ID を設定します。

（注） 固定 IPv4 アドレスをルータ ID として使用することを推奨します。

ステップ 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*]

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# redistribute bgp 100
```

または

```
RP/0/RP0/cpu 0: router(config-router)#redistribute bgp 110
```

1 つのルーティング ドメインから別のルーティング ドメインへの OSPF ルートの再配布

または

あるルーティング ドメインから別のルーティング ドメインへOSPFv3 ルートを再配布します。

- このコマンドを実行すると、定義上ルータが ASBR になります。
- OSPF は再配布によって学習したすべてのルートを external とタグ付けします。
- プロトコルとそのプロセス ID（設定されている場合）は、OSPF に再配布されるプロトコルを示します。
- メトリックは外部ルートに割り当てるコストです。すべてのプロトコルでデフォルトは20です。ただし、BGP のデフォルトのメトリックは 1 です。
- OSPF の例では、BGP 自律システム 1、レベル 1 のルートを OSPF にタイプ 2 外部ルートとして再配布します。
- OSPFv3 の例では、BGP 自律システム 1、レベル 1 および 2 のルートを OSPF に再配布します。OSPFv3 ルーティング ドメインにアドバタイズされるデフォルト ルートに関連付けられている外部リンク タイプは、タイプ 1 の外部ルートです。

（注） OSPFv3 では RPL はサポートされていません。

ステップ 5 次のいずれかを実行します。

- **summary-prefix** *address mask* [**not-advertise**] [**tag** *tag*]
- **summary-prefix** *ipv6-prefix / prefix-length* [**not-advertise**] [**tag** *tag*]

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# summary-prefix 10.1.0.0 255.255.0.0
```

または

```
RP/0/RP0/cpu 0: router(config-router)# summary-prefix 2010:11:22::/32
```

（任意）OSPF の集約アドレスを作成します。

または

（任意）OSPFv3 の集約アドレスを作成します。

- このコマンドは、非 OSPF ルートの外部ルート集約を行います。
- 集約される外部範囲は隣接している必要があります。異なる 2 台のルータからの重複範囲を集約すると、誤った宛先にパケットが送信される原因となる場合があります。
- このコマンドはオプションです。これを指定しないと、各ルートはリンクステートデータベースに含まれ、LSA にアドバタイズされます。
- OSPFv2 の例では、集約アドレス 10.1.0.0 にアドレス 10.1.1.0、10.1.2.0、10.1.3.0 などが含まれています。外部の LSA では、アドレス 10.1.0.0 だけがアドバタイズされます。
- OSPFv3 の例では、集約アドレス 2010:11:22::/32 には 2010:11:22:0:1000::1、2010:11:22:0:2000:679:1、などのアドレスがあります。外部 LSA にはアドレス 2010:11:22::/32 だけがアドバタイズされます。

ステップ 6 commit

例

次の例では、プレフィックス リストを使用して、他のプロトコルから再配布されるルートを制限します。

上位 32 ビットの 9898:1000 および 32 から 64 のプレフィックス長を持つルートだけが BGP 42 から再配布されます。このパターンに一致しないルートだけが BGP 1956 から再配布されます。

```
ipv6 prefix-list list1
 seq 10 permit 9898:1000::/32 ge 32 le 64
ipv6 prefix-list list2
 seq 10 deny 9898:1000::/32 ge 32 le 64
 seq 20 permit ::/0 le 128
router ospfv3 1
 router-id 10.0.0.217
 redistribute bgp 42
 redistribute bgp 1956
 distribute-list prefix-list list1 out bgp 42
 distribute-list prefix-list list2 out bgp 1956
 area 1
  interface TenGigE 0/0/0/0
```

OSPF Version 2 のノンストップ フォワーディング

OSPF Version 2 の NSF では、フェールオーバー後にルーティング プロトコル情報を保存しながら、既知のルートを通ってデータ パケットの転送が継続されるようにできます。NSF を使用すると、ピア ネットワーキング デバイスでルーティング フラップが発生しません。フェールオーバー中、データ トラフィックはインテリジェント ラインカードを介して転送されますが、スタンバイ ルート プロセッサ（RP）では、障害が発生した RP からの制御と見なされます。フェールオーバー中にラインカードのアップ状態が維持され、アクティブ RP の転送情報ベース（FIB）が最新状態に維持される機能が、NSF の動作にとって非常に重要です。

OSPF などのルーティング プロトコルは、アクティブ RP または DRP 上でのみ実行され、隣接ルータからルーティング アップデートを受信します。OSPF NSF 対応ルータが RP のフェールオーバーを実行する場合、リンク ステート データベースを OSPF ネイバーと再同期するために、次の 2 つの処理を実行する必要があります。まず、ネイバー関係をリセットせずに、ネットワーク上の使用可能な OSPF ネイバーを再学習します。次に、ルータはネットワークのリンクステート データベースのコンテンツを再取得します。

RP フェールオーバーの後できるだけ早く、NSF 対応ルータは OSPF NSF 信号を隣接する NSF 対応デバイスに送信します。この信号はフェールオーバー ルータで生成されたリンクローカル LSA の形式になります。ネイバー ネットワーキング デバイスは、この信号をこのルータとのネイバー関係がリセットされるべきでないことを示す指示として認識します。NSF 対応ルータがネットワーク上の他のルータから信号を受信すると、ネイバー リストの再構築を始めます。

ネイバー関係が再構築されると、NSF 対応ルータはすべての NSF 認識ネイバーとデータベースの再同期化を始めます。この時点でルーティング情報は OSPF ネイバーの間で交換されます。交換が完了すると、NSF 対応デバイスはルーティング情報を使用して、失効ルートを削除し、RIB を更新して、新しい転送情報で FIB を更新します。ルータおよび OSPF ネイバー上の OSPF が完全にコンバージされるようになりました。

Cisco for OSPF Version 2 固有のノンストップ フォワーディングの設定

このタスクでは、NSF 対応ルータの Cisco に専用の OSPF NSF を設定する方法を説明します。このタスクはオプションです。

始める前に

OSPF NSF では、すべてのネイバー ネットワーキング デバイスが NSF 対応である必要があります。ルータにソフトウェア イメージをインストールすると自動的に NSF 対応になります。NSF 対応ルータが特定のネットワーク セグメントで NSF 非認識ネイバーを検出すると、そのセグメントで NSF 機能をディセーブルにします。NSF 対応または NSF 認識ルータで完全に構成された他のネットワーク セグメントに対しては、継続して NSF 機能を提供します。

（注） ノンストップ フォワーディングの設定では、次の制約事項が適用されます。

- 仮想リンク用 Cisco OSPF NSF はサポートされません。
- ネイバーは NSF 対応である必要があります。

手順

ステップ 1 **configure**

ステップ 2 **router ospf** *process-name*

例：

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

指定したルーティング プロセスにOSPFルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **router-id** { *router-id* }

例：

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

OSPF プロセスのルータ ID を設定します。

（注） 固定 IPv4 アドレスをルータ ID として使用することを推奨します。

ステップ 4 次のいずれかを実行します。

- **nsf cisco**
- **nsf cisco enforce global**

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# nsf cisco enforce global
```

OSPF プロセスの Cisco NSF 操作をイネーブルにします。

- オプションの **enforce** および **global** キーワードを指定せずに **nsf cisco** コマンドを使用して、検出された NSF 以外のネイバーのインターフェイスで NSF 再起動メカニズムを中断し、NSF ネイバーが適切に機能できるようにします。
- 再起動中にルータが NSF を実行するようにする場合は、オプションの **enforce** および **global** キーワードを指定して **nsf cisco** コマンドを使用します。ただし、NSF 以外のネイバーが検出されると、OSPF プロセス全体で NSF 再起動はキャンセルされます。

ステップ 5 **nsf interval** *seconds*

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# nsf interval 120
```

NSF 再起動の試行間隔の最小時間を設定します。

（注） このコマンドを使用する場合、OSPF が NSF 再起動実行を試みる前の OSPF プロセスを、最小でも 90 秒に設定する必要があります。

ステップ 6 **nsfflush-delay-time***seconds*

例：

```
RP/0/RP0/cpu 0: router(config-ospf)#nsf flush-delay-time 1000
```

外部ルートの学習に許可される最大時間を秒単位で設定します。

ステップ 7 **nsflifetime***seconds*

例：

```
RP/0/RP0/cpu 0: router(config-ospf)#nsf lifetime 90
```

再起動に続く NSF のルートの最大有効期間を秒単位で設定します。

ステップ 8 **nsfietf**

例：

```
RP/0/RP0/cpu 0: router(config-ospf)#nsf ietf
```

ietf グレースフル リスタートをイネーブルにします。

ステップ 9 **commit**

OSPF Shortest Path First スロットリング

OSPF SPF スロットリングにより、SPF スケジューリングをミリ秒間隔で設定して、ネットワークが不安定な場合に SPF 計算を遅らせることができます。トポロジ変化が発生した場合、Shortest Path Tree（SPT）を再計算するように SPF がスケジューリングされます。SPF が 1 回実行されると、複数のトポロジ変化イベントが発生します。

SPF 計算の実行間隔は、ネットワークのトポロジ変化の頻度に応じて動的に選択されます。ユーザ指定値の範囲内で、間隔は選択されます。ネットワークトポロジが不安定な場合、トポロジが安定するまで、SPF スロットリング機能は SPF スケジューリング間隔を長目に計算します。

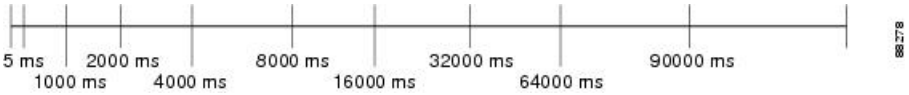
SPF の計算は、timers throttle spf コマンドで設定した間隔で実行されます。待機期間とは、次の SPF 計算が実行されるまで待機する時間のことです。計算を行うたびに、待機期間はその前の期間の 2 倍の長さになり、指定された最大待機期間に達するまでそれが行われます。

SPF タイミングについて、例を使用して説明します。この例では、開始時の間隔は 5 ミリ秒（ms）、初回待機時間は 1000 ミリ秒、最大待機期間は 90,000 ミリ秒に設定されます。

```
timers spf 5 1000 90000
```

図 *2 : timers spf* コマンドで設定される *SPF* の計算間隔

次の図に、ある待機期間中に少なくとも 1 回のトポロジ変化イベントを受信する場合の、SPF 計算の実行間隔を示します。

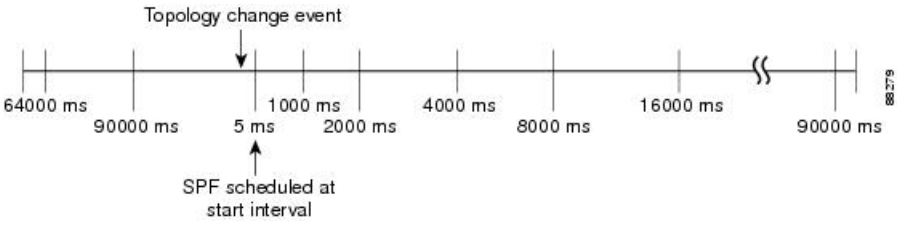


前の待機期間中に少なくとも 1 回のトポロジ変化イベントを受信すると、SPF 計算の待機期間が 2 倍になることに注意してください。最大待機期間に達すると、トポロジが安定し、待機期間中にイベントを受信しなくなるまで、待機期間が変化しなくなります。

現在の待機期間の経過後に、最初のトポロジ変化イベントを受信した場合は、開始時待機期間として指定されている時間だけ SPF 計算が遅延されます。その後の待機期間は、動的パターンに従います。

最大待機期間の開始後に、最初のトポロジ変化イベントが発生した場合、SPF 計算は開始時待機期間で再びスケジューリングされ、その後の待機期間は timers throttle spf コマンドで指定されたパラメータに従ってリセットされます。図 3 : トポロジ変化イベント後のタイマー間隔のリセット （75 ページ） では、最大待機期間の開始後にトポロジ変化イベントを受信して、SPF 間隔がリセットされることに注意してください。

図 *3 :* トポロジ変化イベント後のタイマー間隔のリセット



OSPF Shortest Path First スロットリングの設定

このタスクでは、SPF スケジューリングをミリ秒間隔で設定し、ネットワークが不安定な場合に SPF 計算を遅らせる方法について説明します。このタスクはオプションです。

手順

ステップ 1 **configure**

ステップ 2 次のいずれかを実行します。

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

例：

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

または

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

指定したルーティング プロセスにOSPFルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

または

指定したルーティング プロセスに OSPFv3 ルーティングをイネーブルにし、router ospfv3 コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **router-id** { *router-id* }

例：

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

OSPF プロセスのルータ ID を設定します。

（注） 固定 IPv4 アドレスをルータ ID として使用することを推奨します。

ステップ 4 **timers throttle spf** *spf-start spf-hold spf-max-wait*

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# timers throttle spf 10 4800 90000
```

SPF スロットリング タイマーを設定します。

ステップ 5 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始し、バックボーン エリアを設定します。

- *area-id* 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。IPv4 アドレス形式を使用することを推奨します。

ステップ 6 **interface** *type interface-path-id*

例：

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

インターフェイス コンフィギュレーション モードを開始して、1 つ以上のインターフェイスをエリアに関連付けます。

ステップ 7 **commit**

ステップ 8 次のいずれかを実行します。

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

例：

```
RP/0/RP0/cpu 0: router# show ospf 1
```

または

```
RP/0/RP0/cpu 0: router# RP/0/RP0/CPU0:router# show ospfv3 2
```

（任意）SPF スロットリング タイマーを表示します。

OSPFv3 のグレースフル リスタート

OSPFv3 グレースフル シャットダウン機能により、次の状況でもデータ プレーン機能を維持することができます。

- ソフトウェアのアップグレードまたはダウングレードに伴う再起動などの、計画された OSPFv3 プロセスの再起動
- プロセスのクラッシュに伴う再起動などの、予期しない OSPFv3 プロセスの再起動

また、プロセッサが使用可能なメモリで非常に低いことを示す重大なメモリ イベントが sysmon のウォッチ ドッグ プロセスから受信された場合、OSPFv3 は一方的にシャットダウンするか、または終了状態に入ります。

この機能を使うと、OSPFv3 ルーティング プロトコルが再起動している間に、確立されているルートでノンストップ データ転送が行われます。そのため、この機能により IPv6 転送の可用性が向上します。

OSPFv3 グレースフル リスタートの設定

このタスクでは、OSPFv3 プロセスのグレースフル リスタートを設定する方法を説明します。このタスクはオプションです。

手順

ステップ 1 **configure**

ステップ 2 **router ospfv3** *process-name*

例：

```
RP/0/RP0/cpu 0: router(config)# router ospfv3 test
```

OSPFv3 のルータ コンフィギュレーション モードを開始します。プロセス名は OSPF ルーティング プロセスを一意に識別する 1 つの単語です。プロセス名はスペースを含まない 40 文字以内の任意の英数字ストリングです。

ステップ 3 **graceful-restart**

例：

```
RP/0/RP0/cpu 0: router(config-ospfv3)#graceful-restart
```

現行ルータでグレースフル リスタートをイネーブルにします。

ステップ 4 **graceful-restart lifetime**

例：

```
RP/0/RP0/cpu 0: router(config-ospfv3)# graceful-restart lifetime 120
```

グレースフル リスタートの最大時間を指定します。

- デフォルトは 95 秒です。
- 値の範囲は 90 ～ 3600 秒です。

ステップ 5 **graceful-restart interval** *seconds*

例：

```
RP/0/RP0/cpu 0: router(config-ospfv3)# graceful-restart interval 120
```

現行ルータのグレースフル リスタートの間隔（最小時間）を指定します。

- 間隔のデフォルト値は 90 秒です。
- 値の範囲は 90 ～ 3600 秒です。

ステップ 6 **graceful-restart helper disable**

例：

```
RP/0/RP0/cpu 0: router(config-ospfv3)# graceful-restart helper disable
```

ヘルパー機能をディセーブルにします。

ステップ 7 **commit**

ステップ 8 **show ospfv3** [*process-name* [*area-id*]] **database grace**

例：

```
RP/0/RP0/cpu 0: router# show ospfv3 1 database grace
```

グレースフル リスタート リンクのステートを表示します。

グレースフル リスタートに関する情報の表示

ここでは、グレースフル リスタートに関する情報を表示するために使用できるタスクについて説明します。

- 機能がイネーブルかどうかや、グレースフル リスタートが最後に実行された時間を確認するには、show ospf コマンドを使用します。OSPFv3 インスタンスの詳細を参照するには、**show ospfv3** *process-name* [*area-id*] **database grace** コマンドを使用します。

グレースフル リスタート機能のステータスの表示

次の画面出力は、ローカル ルータのグレースフル リスタート機能の状態を示しています。

```
RP/0/RP0/cpu 0: router# show ospfv3 1 database grace

 Routing Process "ospfv3 1" with ID 2.2.2.2
 Initial SPF schedule delay 5000 msecs
 Minimum hold time between two consecutive SPFs 10000 msecs
 Maximum wait time between two consecutive SPFs 10000 msecs
 Initial LSA throttle delay 0 msecs
 Minimum hold time for LSA throttle 5000 msecs
 Maximum wait time for LSA throttle 5000 msecs
 Minimum LSA arrival 1000 msecs
 LSA group pacing timer 240 secs
 Interface flood pacing timer 33 msecs
 Retransmission pacing timer 66 msecs
 Maximum number of configured interfaces 255
 Number of external LSA 0. Checksum Sum 00000000
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
 Graceful Restart enabled, last GR 11:12:26 ago (took 6 secs)
    Area BACKBONE(0)
        Number of interfaces in this area is 1
        SPF algorithm executed 1 times
        Number of LSA 6. Checksum Sum 0x0268a7
        Number of DCbitless LSA 0
        Number of indication LSA 0
        Number of DoNotAge LSA 0
        Flood list length 0
```

OSPFVersion2のウォームスタンバイとノンストップルーティング

OSPFv2 ウォーム スタンバイは、RP のスイッチオーバー全体でハイ アベイラビリティを実現します。ウォーム スタンバイ拡張機能により、アクティブ RP で実行されているプロセスごとに、スタンバイ RP で開始された、対応するスタンバイ プロセスがあります。スタンバイ OSPF プロセスは、アクティブな OSPF プロセスにパフォーマンスへ影響を与えることなく、OSPF パケットを送受信できます。

ノンストップ ルーティング（NSR）によって、RP フェールオーバー、プロセスの再起動、またはインサービス アップグレードはピア ルータから見えなくなり、パフォーマンスまたは処理への影響が最小限になります。ルーティング プロトコルはルータ間でやり取りされるため、NSR の影響を受けません。NSR はウォーム スタンバイ拡張機能によって構築されます。NSR を使用すると Cisco NSF および IETF グレースフル リスタート プロトコル拡張機能の要件が緩和されます。

OSPFv2 のノンストップ ルーティングのイネーブル化

このオプションのタスクでは、OSPFv2 プロセスのノンストップ ルーティング（NSR）をイネーブルにする方法を説明します。NSR はデフォルトでディセーブルになっています。NSR をイネーブルにすると、アクティブな RP の OSPF プロセスは必要なすべてのデータと状態をスタンバイ RP の OSPF プロセスと同期します。スイッチオーバーが発生すると、新たにアクティブになった RP の OSPF プロセスには実行の継続に必要なすべてのデータと状態が含まれており、ネイバーからの援助は必要ありません。

手順

ステップ 1 **configure**

ステップ 2 **router ospf** *instance-id*

例：

```
RP/0/RP0/cpu 0: router(config)# router ospf isp
```

指定したルーティング プロセスに OSPF ルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。この例では、OSPF インスタンスは isp と呼ばれます。

ステップ 3 **nsr**

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# nsr
```

OSPFv2 プロセスの NSR をイネーブルにします。

ステップ 4 **commit**

OSPFVersion3のウォームスタンバイとノンストップルーティング

この機能を使うと、フェールオーバー（FO）の前にOSPFv3が自動で初期化され、障害が発生する前に機能する準備が整います。また、スイッチオーバー中のダウンタイムを減らすことができます。デフォルトでは、ルータは hello パケットを 40 秒ごとに送信します。

各 OSPF プロセスのウォーム スタンバイ プロセスが、アクティブ ルート プロセッサで実行されている場合、対応する OSPF プロセスはスタンバイ RP で開始する必要があります。この機能のためにコンフィギュレーションを変更する必要はありません。

ウォーム スタンバイは常にイネーブルです。この機能は、IGP として OSPFv3 を実行しているシステムが RP フェールオーバーを実行するときに有利です。

OSPFv3 のノンストップ ルーティングのイネーブル化

このタスクでは、OSPFv3 プロセスのノンストップ ルーティング（NSR）をイネーブルにする方法を説明します。NSR はデフォルトでディセーブルになっています。NSR をイネーブルにすると、アクティブな RP の OSPF プロセスは必要なすべてのデータと状態をスタンバイ RP の OSPF プロセスと同期します。スイッチオーバーが発生すると、新たにアクティブになった RP の OSPF プロセスには実行の継続に必要なすべてのデータと状態が含まれており、ネイバーからの援助は必要ありません。

手順

ステップ 1 **configure**

ステップ 2 **router ospfv3** *instance-id*

例：

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

指定したルーティング プロセスにOSPFルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。この例では、OSPF インスタンスは isp と呼ばれます。

ステップ 3 **nsr**

例：

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

OSPFv3 プロセスの NSR をイネーブルにします。

ステップ 4 commit

OSPFv2OSPF SPF プレフィックスのプライオリティ設定

OSPFv2OSPF SPF のプレフィックスのプライオリティ設定機能によって、ルートのインストール中に、高速モードで、管理者が重要なプレフィックスを収束できます。

多くのプレフィックスがルーティング情報ベース（RIB）および転送情報ベース（FIB）にインストールされる必要がある場合、SPF 中の、最初のプレフィックスから最後のプレフィックスまでの更新期間が、かなりの長さになることがあります。

時間依存のトラフィック（VoIP など）が他のトラフィック フローとともに同じルータを通過する可能性があるネットワークでは、SPF 中の、これらの時間に依存するプレフィックスの RIB および FIB アップデートを優先することが重要です。

OSPFv2OSPF SPF のプレフィックスのプライオリティ設定機能によって、SPF 計算中に RIB にインストールされる重要なプレフィックスに、管理者が優先順位を付けることが可能になります。重要なプレフィックスは、領域ごとに同じルート タイプのプレフィックス内で高速で収束します。RIB および FIB のインストール前に、ルートとプレフィックスは指定したルート ポリシーに基づいて OSPF ローカル RIB のさまざまなプライオリティ バッチ キューに割り当てられます。RIB プライオリティ バッチ キューはプライオリティの高い順から「critical」、「high」、「medium」、「low」に分類されます。

イネーブルの場合、次のプレフィックス プライオリティで RIB 更新シーケンスが変更されます。

Critical > High > Medium > Low

プレフィックス プライオリティが設定されると、デフォルトでは /32 プレフィックスは優先されなくなり、より高いプライオリティ ポリシーに一致しない場合は、low プライオリティ キューに配置されます。ルート ポリシーは、/32 が高いプライオリティのキュー（High プライオリティ、または Medium プライオリティ）に保持されるように考案する必要があります。

プライオリティはルート ポリシーを使用して指定されます。このルート ポリシーは、IP アドレスまたはルート タグに基づいて照会することができます。SPF 中に、指定したルート ポリシーに対してプレフィックスがチェックされ、適切な RIB バッチ プライオリティ キューに割り当てられます。

これらは、このシナリオの例です。

- high プライオリティ ルート ポリシーだけを指定した場合は、medium プライオリティに対してルート ポリシーは設定されません。
  - 許可されたプレフィックスは、high プライオリティ キューに配置されます。
  - /32 を含む一致しないプレフィックスは、low プライオリティ キューに配置されます。
- high プライオリティと medium プライオリティの両方のルート ポリシーが指定され、critical プライオリティにマップが指定されない場合

- highプライオリティのルートポリシーに一致する許可されたプレフィックスは、high プライオリティ キューに配置されます。
- medium プライオリティのルート ポリシーに一致する許可されたプレフィックスは、medium プライオリティ キューに配置されます。
- /32 を含む一致しないプレフィックスは、lowプライオリティ キューに移動されます。

- criticalプライオリティと highプライオリティの両方のルート ポリシーが指定されており、medium プライオリティにマップが指定されていない場合
  - critical プライオリティのルート ポリシーに一致する許可されたプレフィックスは、critical プライオリティ キューに配置されます。
  - highプライオリティのルートポリシーに一致する許可されたプレフィックスは、high プライオリティ キューに配置されます。
  - /32 を含む一致しないプレフィックスは、lowプライオリティ キューに配置されます。

- mediumプライオリティ ルートポリシーだけが指定され、highプライオリティまたはcritical プライオリティにマップが指定されていない場合
  - medium プライオリティのルート ポリシーに一致する許可されたプレフィックスは、medium プライオリティ キューに割り当てられます。
  - /32 を含む一致しないプレフィックスは、lowプライオリティ キューに配置されます。

[no] spf prefix-priority route-policy *rpl* コマンドを使用して、SPF 中に OSPFv2OSPF プレフィックス インストールのプライオリティをグローバル RIB で設定します。

SPF プレフィックスのプライオリティ設定は、デフォルトではディセーブルです。ディセーブル モードでは、/32 プレフィックスは他のプレフィックスよりも前にグローバル RIB にインストールされます。SPF プライオリティ設定がイネーブルの場合、ルートは route-policy 基準に対して照会され、SPF プライオリティ セットに基づいて適切なプライオリティ キューに割り当てられます。/32 を含む一致しないプレフィックスは、low プライオリティのキューに配置されます。

すべての /32 を high プライオリティ キューまたは medium プライオリティ キューで処理する必要がある場合、次の 1 つのルート マップを設定します。

```
prefix-set ospf-medium-prefixes
  0.0.0.0/0 ge 32
  end-set
```

OSPFv2 OSPF SPF プレフィックス プライオリティの設定

このタスクを実行して、OSPFv2 OSPF SPF（Shortest Path First）プレフィックス プライオリティを設定します。

手順

ステップ 1 **configure**

ステップ 2 **prefix-set** *prefix-set name*

例：

```
RP/0/RP0/cpu 0: router(config)#prefix-set ospf-critical-prefixes
RP/0/RP0/cpu 0: router(config-pfx)#66.0.0.0/16
RP/0/RP0/cpu 0: router(config-pfx)#end-set
```

プレフィックス セットを設定します。

ステップ 3 **route-policy** *route-policy name* **if destination in** *prefix-set name* **then set spf-priority** {**critical** | **high** | **medium**} **endif**

例：

```
RP/0/RP0/cpu 0: router#route-policy ospf-spf-priority
RP/0/RP0/cpu 0: router(config-rpl)#if destination in ospf-critical-prefixes then
 set spf-priority critical
endif
RP/0/RP0/cpu 0: router(config-rpl)#end-policy
```

ルート ポリシーと OSPF SPF プライオリティを設定します。

ステップ 4 次のいずれかのコマンドを使用します。

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

例：

```
RP/0/RP0/cpu 0: router# router ospf 1
```

または

```
RP/0/RP0/cpu 0: router# router ospfv3 1
```

ルータ OSPF コンフィギュレーション モードを開始します。

ステップ 5 **router ospf** *ospf name*

例：

```
RP/0/RP0/cpu 0: router# router ospf 1
```

ルータ OSPF コンフィギュレーション モードを開始します。

ステップ 6 **spf prefix-priority route-policy** *route-policy name*

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# spf prefix-priority route-policy ospf-spf-priority
```

または

```
RP/0/RP0/cpu 0: router(config-ospfv3)#spf prefix-priority route-policy ospf3-spf-priority
```

定義されているルート ポリシーの SPF プレフィックス プライオリティを設定します。

（注） OSPF ルータで spf prefix-priority コマンドを設定します。

ステップ 7 **commit**

ステップ 8 **show rpl route-policy** *route-policy name* **detail**

例：

```
RP/0/RP0/cpu 0: 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
  !
```

SPF のプレフィックス プライオリティのセットを表示します。

OSPFv2

OSPFv3

この例では、/32 プレフィックスを一般的に medium プライオリティに設定し、一部の /32 および /24 プレフィックスを critical プライオリティおよび high プライオリティ キューに設定する方法を示します。

```
prefix-set ospf-critical-prefixes
  192.41.5.41/32,
  11.1.3.0/24,
  192.168.0.44/32
  end-set
  !
  prefix-set ospf-high-prefixes
  44.4.10.0/24,
  192.41.4.41/32,
  41.4.41.41/32
  end-set
  !
  prefix-set ospf-medium-prefixes
  0.0.0.0/0 ge 32
  end-set
  !

  route-policy ospf-priority
    if destination in ospf-high-prefixes then
      set spf-priority high
    else
      if destination in ospf-critical-prefixes then
```

```
        set spf-priority critical
      else
        if destination in ospf-medium-prefixes then
          set spf-priority medium
        endif
      endif
    endif
  end-policy

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

router ospfv3 1
   spf prefix-priority route-policy ospf-priority
   area 0
    interface TenGigE 0/0/0/1
    !
   !
   area 3
    interface TenGigE 0/0/0/0
    !
   !
   area 8
    interface TenGigE 0/0/0/0
```

プロバイダー エッジからカスタマー エッジ（PE-CE）プロトコルとしての OSPF の設定

手順

ステップ 1 **configure**

ステップ 2 **router ospf** *process-name*

例：

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

指定したルーティング プロセスにOSPF ルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **vrf** *vrf-name*

例：

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

VRF インスタンスを作成し、VRF コンフィギュレーション モードを開始します。

ステップ 4 **router-id** { *router-id* }

例：

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

OSPF プロセスのルータ ID を設定します。

（注） 固定 IPv4 アドレスをルータ ID として使用することを推奨します。

ステップ 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*]

例：

```
RP/0/RP0/cpu 0: router(config-ospf-vrf)# redistribute bgp 1 level-1
```

1 つのルーティング ドメインから別のルーティング ドメインへの OSPF ルートの再配布

- このコマンドを実行すると、定義上ルータが ASBR になります。
- OSPF は再配布によって学習したすべてのルートを external とタグ付けします。
- プロトコルとそのプロセス ID（設定されている場合）は、OSPF に再配布されるプロトコルを示します。
- メトリックは外部ルートに割り当てるコストです。すべてのプロトコルでデフォルトは20です。ただし、BGP のデフォルトのメトリックは 1 です。
- この例では、BGP 自律システム 1、レベル 1 のルートを OSPF にタイプ 2 の外部ルートとして再配布します。

ステップ 6 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始し、OSPF プロセスのエリアを設定します。

- *area-id* 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。

ステップ 7 **interface** *type interface-path-id*

例：

```
RP/0/RP0/cpu 0: router(config-ospf-vrf)# interface TenGigE 0/0/0/0
```

インターフェイス コンフィギュレーション モードを開始して、1 つ以上のインターフェイスを VRF に関連付けます。

ステップ 8 exit

例：

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

インターフェイス コンフィギュレーション モードを終了します。

ステップ 9 **domain-id** *[* **secondary** *]* **type** *{* **0005** | **0105** | **0205** | **8005** *}* **value** *value*

例：

```
RP/0/RP0/cpu 0: router(config-ospf-vrf)# domain-id type 0105 value 1AF234
```

OSPF VRF のドメイン ID を指定します。

- *value* 引数は 6 オクテットの 16 進数です。

ステップ 10 **domain-tag** *tag*

例：

```
RP/0/RP0/cpu 0: router(config-0spf-vrf)# domain-tag 234
```

OSPF VRF ドメイン タグを指定します。

- *tag* の有効範囲は、0 ～ 4294967295 です。

ステップ 11 disable-dn-bit-check

例：

```
RP/0/RP0/cpu 0: router(config-ospf-vrf)# disable-dn-bit-check
```

ダウン ビットを無視するように設定します。

ステップ 12 **commit**

複数の OSPF インスタンスの作成（OSPF プロセスおよび VRF）

このタスクでは、複数の OSPF インスタンスの作成方法を説明します。この場合、インスタンスは通常の OSPF インスタンスと VRF インスタンスです。

手順

ステップ 1 **configure**

ステップ 2 **router ospf** *process-name*

例：

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

指定したルーティング プロセスにOSPFルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始し、バックボーン エリアを設定します。

- *area-id* 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。IPv4 アドレス形式を使用することを推奨します。

ステップ 4 **interface** *type interface-path-id*

例：

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

インターフェイス コンフィギュレーション モードを開始して、1 つ以上のインターフェイスをエリアに関連付けます。

ステップ 5 **exit**

例：

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

OSPF コンフィギュレーション モードを開始します。

ステップ 6 **vrf** *vrf-name*

例：

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

VRF インスタンスを作成し、VRF コンフィギュレーション モードを開始します。

ステップ 7 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始して、OSPF プロセスで VRF インスタンスのエリアを設定します。

- *area-id* 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。

ステップ 8 **interface** *type interface-path-id*

例：

```
RP/0/RP0/cpu 0: router(config-ospf-vrf)# interface TenGigE 0/0/0/0
```

インターフェイス コンフィギュレーション モードを開始して、1 つ以上のインターフェイスを VRF に関連付けます。

ステップ 9 **commit**

OSPF のラベル配布プロトコル IGP 自動設定

ラベル配布プロトコル（LDP）内部ゲートウェイ プロトコル（IGP）自動設定を使うと、OSPF などの IGP インスタンスに使用されているインターフェイスのセットで LDP をイネーブルにする手順を簡略化できます。LDP IGP 自動設定は、多数のインターフェイス（転送に LDP がコアで使用される場合など）および複数の OSPF インスタンスで同時に使用できます。

この機能は、デフォルトの VPN ルーティングおよび転送（VRF）インスタンスとして IPv4 ユニキャスト アドレス ファミリをサポートします。

LDP IGP 自動設定は、LDP の個々のインターフェイス ベースで igp auto-config disable コマンド使用して明示的にディセーブルにすることもできます。これにより、明示的にディセーブルにしたインターフェイスを除くすべての OSPF インターフェイスを LDP で受信できます。

OSPF のラベル配布プロトコル IGP 自動設定の設定

このタスクでは、OSPF インスタンスに対する LDP 自動設定を設定する方法について説明します。

オプションで、OSPF インスタンスのエリアにこの機能を設定できます。

手順

ステップ 1 **configure**

ステップ 2 **router ospf** *process-name*

例：

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

指定したルーティングプロセスにOSPFルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **mpls ldp auto-config**

例：

```
RP/0/RP0/cpu 0: router(config-ospf)#  mpls ldp auto-config
```

OSPF インスタンスの LDP IGP インターフェイスの自動設定をイネーブルにします。

- 任意で、このコマンドを OSPF インスタンスのエリアに設定されます。

ステップ 4 **commit**

LDP IGP 同期の設定：OSPF

このタスクを実行して、OSPF で LDP IGP 同期を設定します。

（注） デフォルトでは、LDP と IGP 間の同期は行われません。

手順

ステップ 1 **configure**

ステップ 2 **router ospf** *process-name*

例：

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

OSPF ルーティング プロセスを識別し、OSPF コンフィギュレーション モードを開始します。

ステップ 3 （任意） **vrf** *vrf-name*

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# vrf red
```

デフォルト以外の VRF を指定します。

ステップ 4 次のいずれかのコマンドを使用します。

- **mpls ldp sync**

- **area** *area-id* **mpls ldp sync**
- **area** *area-id* **interface** *name* **mpls ldp sync**

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# mpls ldp sync
```

インターフェイスで LDP IGP 同期をイネーブルにします。

ステップ 5 （任意） 次のいずれかのコマンドを使用します。

- **mpls ldp sync**
- **area** *area-id* **mpls ldp sync**
- **area** *area-id* **interface** *name* **mpls ldp sync**

例：

```
RP/0/RP0/cpu 0: router(config-ospf-vrf)# mpls ldp sync

RP/0/RP0/cpu 0: router(config-ospf-vrf)# area 1 mpls ldp sync
```

指定された VRF のインターフェイス上で LDP IGP 同期を有効にします。

ステップ 6 **commit**

ステップ 7 （任意） **show mpls ldp vrf** *vrf-name* **ipv4 igp sync**

例：

```
RP/0/RP0/cpu 0: router# show mpls ldp vrf red ipv4 igp sync
```

アドレス ファミリ IPv4 の指定された VRF に関する LDP IGP 同期情報を表示します。

ステップ 8 （任意） **show mpls ldp vrf all ipv4 igp sync**

例：

```
RP/0/RP0/cpu 0: router# show mpls ldp vrf all ipv4 igp sync
```

アドレス ファミリ IPv4 のすべての VRF に関する LDP IGP 同期情報を表示します。

ステップ 9 （任意） **show mpls ldp** { **ipv4** | **ipv6** }**igp sync**

例：

```
RP/0/RP0/cpu 0: router# show mpls ldp ipv4 igp sync

RP/0/RP0/cpu 0: router# show mpls ldp ipv6 igp sync
```

IPv4 または IPv6 アドレス ファミリの LDP IGP 同期情報を表示します。

例

次に、OSPF の LDP IGP 同期を設定する例を示します。

```
router ospf 100
mpls ldp sync
!
mpls ldp
 igp sync delay 30
!
```

OSPF 認証のメッセージ ダイジェスト管理

すべてのOSPFルーティングプロトコル交換は認証されます。使用される方法は、認証が設定される方法によって異なります。暗号認証を使用する場合、OSPF ルーティング プロトコルは、Message Digest 5（MD5）認証アルゴリズムを使用してネットワーク内のネイバー間で送信されたパケットを認証します。各 OSPF プロトコル パケットでは、キーを使用して、OSPF パケットの最後に付加されるメッセージ ダイジェストを生成および検証します。メッセージ ダイジェストはOSPFプロトコルパケットおよび秘密キーの単方向機能です。各キーは使用されるインターフェイスとキー ID の組み合わせで識別されます。インターフェイスでは、複数のキーが常にアクティブになっています。

キーのロールオーバーを管理し、OSPFのMD5認証を拡張するには、キーチェーンと呼ばれるキーのコンテナを設定できます。この各キーは、生成/受け取り時間、キー ID、認証アルゴリズムの属性で構成されます。

OSPF の認証メッセージ ダイジェスト管理の設定

このタスクでは、OSPFインターフェイスのキーチェーンの認証を管理する方法を説明します。

始める前に

このタスクを実行するには、有効なキーチェーンを設定する必要があります。

手順

ステップ 1 **configure**

ステップ 2 **router ospf** *process-name*

例：

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

指定したルーティングプロセスにOSPFルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **router-id** { *router-id* }

例：

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

OSPF プロセスのルータ ID を設定します。

（注） 固定 IPv4 アドレスをルータ ID として使用することを推奨します。

ステップ 4 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始します。

area-id 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。IPv4 アドレス形式を使用することを推奨します。

ステップ 5 **interface** *type interface-path-id*

例：

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

インターフェイス コンフィギュレーション モードを開始して、1 つ以上のインターフェイスをエリアに関連付けます。

ステップ 6 **authentication** [**message-digest** *keychain* | **keychain** *keychain* | **null**]

MD5 キーチェーンを設定します。キーチェーンは、次の 3 つのレベルの認証で設定できます。

- ルータ レベルの認証
- エリア レベルの認証
- インターフェイス レベルの認証

例：

次に、キーチェーン認証の設定例を示します。

```
RP/0/RP0/CPU0:router(config-ospf)# authentication keychain test_chain  -----□ Router-level
 authentication
RP/0/RP0/CPU0:router(config-ospf)# router-id 1.1.1.1
RP/0/RP0/CPU0:router(config-ospf)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-ospf)# area 1
RP/0/RP0/CPU0:router(config-ospf-ar)# authentication keychain test_chain  -----□ Area-level
 authentication
RP/0/RP0/CPU0:router(config-ospf-ar)# interface TenGigE 0/0/0/1
RP/0/RP0/CPU0:router(config-ospf-ar-if)# authentication keychain test_chain ---□
```

```
Interface-level authentication
RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit
```

例：

次に、message-digest 認証の設定例を示します。

```
RP/0/RP0/cpu 0: router(config-ospf-ar-if)# authentication message-digest keychain ospf_int1
```

（注） 上記の例では、この手順を実行する前に*ospf_intl* キーチェーンを設定する必要があります。

ステップ 7 commit

例

次の例は、5 つのキー ID を持つキーチェーン *ospf_intf_1* の設定方法を示します。各キー ID は異なる **send-lifetime** 値で設定されます。ただし、すべてのキー ID はキーに同じテキスト文字列を指定します。

```
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
```

次の例は、TenGigE 0/0/0/0 インターフェイスでキーチェーン認証がイネーブルであることを示します。

```
show ospf 1 interface TenGigE 0/0/0/0

TenGigE 0/0/0/0 is up, line protocol is up
  Internet Address 100.10.10.2/24, Area 0
  Process ID 1, Router ID 2.2.2.1, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 2.2.2.1, Interface address 100.10.10.2
  Backup Designated router (ID) 1.1.1.1, Interface address 100.10.10.1
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:02
```

```
  Index 3/3, flood queue length 0
  Next 0(0)/0(0)
  Last flood scan length is 2, maximum is 16
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 1.1.1.1  (Backup Designated Router)
  Suppress hello for 0 neighbor(s)
  Keychain-based authentication enabled
    Key id used is 3
  Multi-area interface Count is 0
```

次に、アクティブに設定されたキーの出力の例を示します。

```
show key chain ospf_intf_1

 Key-chain: ospf_intf_1/ -

 Key 1 -- text "0700325C4836100B0314345D"
   cryptographic-algorithm -- MD5
   Send lifetime:   11:30:30, 01 May 2007 - (Duration) 600
   Accept lifetime: Not configured
 Key 2 -- text "10411A0903281B051802157A"
   cryptographic-algorithm -- MD5
   Send lifetime:   11:40:30, 01 May 2007 - (Duration) 600
   Accept lifetime: Not configured
 Key 3 -- text "06091C314A71001711112D5A"
   cryptographic-algorithm -- MD5
   Send lifetime:   11:50:30, 01 May 2007 - (Duration) 600  [Valid now]
   Accept lifetime: Not configured
 Key 4 -- text "151D181C0215222A3C350A73"
   cryptographic-algorithm -- MD5
   Send lifetime:   12:00:30, 01 May 2007 - (Duration) 600
   Accept lifetime: Not configured
 Key 5 -- text "151D181C0215222A3C350A73"
   cryptographic-algorithm -- MD5
   Send lifetime:   12:10:30, 01 May 2007 - (Duration) 600
   Accept lifetime: Not configured
```

OSPF の GTSM TTL セキュリティ メカニズム

OSPF は、ネイバーに対するネットワーク、フラッディング リンク ステート アドバタイズメント（LSA）アップデートで、トポロジの変更を検出し、トポロジの新しいビュー上ですばやくコンバージするためにネットワーキング デバイスを必要とするリンク ステート プロトコルです。ただし、ネイバーからの LSA の受信動作中は、ネットワーク攻撃が発生する可能性があります。これは、ユニキャストパケットが仮想リンクの 1 ホップまたは複数ホップ向こう側に配置されているネイバーから送信されているという確認ができないためです。

仮想リンクについては、OSPF パケットはネットワーク全体の複数ホップを通過して送信されます。したがって、TTL 値は複数回にわたり減少していく可能性があります。このようなリンクの種類では、最小 TTL 値が複数ホップ パケットで許可され受け入れられなければなりません。

複数ホップを通過して送信される無効なソースから発生するネットワーク攻撃をフィルタリングするには、一般 TTL セキュリティ メカニズム（GTSM）の RFC 3682 を使用して、攻撃を防

止します。GTSM はリンク ローカル アドレスをフィルタリングして、TTL 値 255 のコンフィギュレーションの 1 ホップ ネイバーとなる隣接関係だけを許可します。IP ヘッダーの TTL 値は OSPF パケットが生成されるときに 255 に設定され、受信された OSPF パケットでデフォルトの GTSM TTL 値 255 またはユーザ設定された GTSM TTL 値に対してチェックされます。このようにして、TTL ホップを超える不正な OSPF パケットをブロックします。

OSPF の一般 TTL セキュリティ メカニズム（GTSM）の設定

このタスクでは、GTSM のインターフェイスにおけるセキュリティの存続可能時間メカニズムの設定方法を説明します。

手順

ステップ 1 **configure**

ステップ 2 **router ospf** *process-name*

例：

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

指定したルーティング プロセスに OSPF ルーティングをイネーブルにし、ルータ コンフィギュレーション モードでルータを配置します。

（注） *process-name* 引数は、40 文字未満の英数字です。

ステップ 3 **router-id** { *router-id* }

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# router id 10.10.10.100
```

OSPF プロセスのルータ ID を設定します。

（注） 固定 IPv4 アドレスをルータ ID として使用することを推奨します。

ステップ 4 **log adjacency changes** [**detail** | **disable**]

例：

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

（任意）ネイバー変更の通知を要求します。

- デフォルトでは、この機能はイネーブルです。
- ネイバー変更によって生成されたメッセージは通知と見なされます。このメッセージは logging console コマンドで重大度レベル 5 に分類されます。logging console コマンドではどの重大度レベルのメッセージをコンソールに送信するかを制御します。デフォルトでは、すべての重大度レベルのメッセージが送信されます。

ステップ 5 **nsf** { **cisco** [**enforce global**] | **ietf** [**helper disable**]}

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# nsf ietf
```

（任意）NSF OSPF プロトコルを設定します。

この例ではグレースフル リスタートをイネーブルにします。

ステップ 6 **timers throttle spf** *spf-start spf-hold spf-max-wait*

例：

```
RP/0/RP0/cpu 0: router(config-ospf)# timers throttle spf 500 500 10000
```

（任意）SPF スロットリング タイマーを設定します。

ステップ 7 **area** *area-id*

例：

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

エリア コンフィギュレーション モードを開始します。

area-id 引数は、area 1000 や area 0.0.3.232 など、ドット付き 10 進表記または IPv4 アドレス形式で入力できます。ただし、1 つのエリアでは同じ形式を選択する必要があります。IPv4 アドレス形式を使用することを推奨します。

ステップ 8 **interface** *type interface-path-id*

例：

```
RP/0/RP0/cpu 0: router(config-ospf-ar)# interface TenGigE0/0/0/0
```

インターフェイス コンフィギュレーション モードを開始して、1 つ以上のインターフェイスをエリアに関連付けます。

ステップ 9 **security ttl** [**disable** | **hops** *hop-count*]

例：

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

OSPF パケットの IP ヘッダーのセキュリティ TTL 値を設定します。

ステップ 10 **commit**

ステップ 11 **show ospf** [*process-name*] [*area-id*] **interface** [*type interface-path-id*]

例：

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

OSPF インターフェイス情報を表示します。

例

OSPF インターフェイスに設定されている GTSM セキュリティ TTL 値を表示する出力例を次に示します。

```
show ospf 1 interface  TenGigE0/5/0/0

 TenGigE0/0/0/0 is up, line protocol is up
   Internet Address 120.10.10.1/24, Area 0
   Process ID 1, Router ID 100.100.100.100, Network Type BROADCAST, Cost: 1
   Transmit Delay is 1 sec, State BDR, Priority 1
   TTL security enabled, hop count 2
   Designated Router (ID) 102.102.102.102, Interface address 120.10.10.3
   Backup Designated router (ID) 100.100.100.100, Interface address 120.10.10.1
   Flush timer for old DR LSA due in 00:02:36
   Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
     Hello due in 00:00:05
   Index 1/1, flood queue length 0
   Next 0(0)/0(0)
   Last flood scan length is 1, maximum is 4
   Last flood scan time is 0 msec, maximum is 0 msec
   Neighbor Count is 1, Adjacent neighbor count is 1
     Adjacent with neighbor 102.102.102.102  (Designated Router)
   Suppress hello for 0 neighbor(s)
   Multi-area interface Count is 0
```

OSPF の参照

OSPF を実装するには、次の概念を理解する必要があります。

OSPF 機能の概要

OSPF は、IP 用のルーティング プロトコルです。これは、ディスタンスベクトル プロトコルではなく、リンクステート プロトコルです。リンクステート プロトコルは、送信元マシンと宛先マシンを接続するリンクの状態に基づいて、ルーティングの決定を行います。リンク ステートは、インターフェイスと、その隣接ネットワーキングデバイスとの関係を説明するものです。インターフェイス情報には、インターフェイスの IP アドレス、ネットワーク マスク、接続されているネットワークの種類、そのネットワークに接続されているルータなどがあります。この情報は、さまざまなタイプのリンクステートアドバタイズメント（LSA）によって伝播します。

ルータは受信したLSAデータの集まりをリンクステートデータベースに格納します。このデータベースにはこのルータのリンクの LSA データが含まれます。ダイクストラ アルゴリズムが採用されている場合、データベースの内容からデータが抽出されてOSPFルーティングテーブルが作成されます。データベースとルーティングテーブルの違いは、データベースにはすべてのrawデータが含まれており、ルーティングテーブルには特定のルータ インターフェイス ポートを介した既知の宛先への最短パスのリストが含まれていることです。

OSPF は大規模ネットワークにまで拡張できるため、IGP として適しています。エリアを使用してネットワークをより管理しやすい大きさに分割するとともに、ネットワークに階層を導入します。ルータはネットワークの 1 つのエリアまたは複数のエリアに接続されます。エリア内のすべてのネットワーキング デバイスは、デバイスが属するエリア内のみのリンク ステートがすべて揃った、同じデータベース情報を維持します。ネットワーク内のすべてのリンク ステートについての情報は持ちません。エリア内のルータ間におけるデータベース情報の合意はコンバージェンスと呼ばれます。

ドメイン内レベルで、OSPF は Intermediate System-to-Intermediate System（IS-IS）を使用して取得したルートを取り込むことができます。OSPF ルートを IS-IS に伝達することもできます。ドメイン間レベルで、OSPF はボーダー ゲートウェイ プロトコル（BGP）を使用して取得したルートを取り込むことができます。OSPF ルートを BGP に伝達することもできます。

Routing Information Protocol（RIP）とは異なり、OSPF は定期的なルーティング アップデートを送信しません。OSPF ルータはネイバーになると、データベースを交換および同期することによって隣接関係を確立します。その後、変更されたルーティング情報だけが伝播されます。エリア内のすべてのルータは自分のリンクのコストとステートをアドバタイズします。この情報は LSA 内で送られます。このステート情報は、1 ホップ先のすべての OSPF ネイバーに送られます。その後すべての OSPF ネイバーは、ステート情報を変更せずに送信します。このフラッディング プロセスは、エリア内のすべてのデバイスが同じリンクステート データベースを持つまで続けられます。

宛先への最適なルートを決定するために、宛先へのルートに含まれるリンクのすべてのコストがソフトウェアによって合計されます。各ルータが別のネットワーキング デバイスからルーティング情報を受信した後で、Shortest Path First（SPF）アルゴリズムが実行されて、データベース内の各宛先ネットワークへの最適なパスが計算されます。

OSPF を実行しているネットワーキング デバイスは、ネットワーク内のトポロジの変化を検出して、リンクステート アップデートをネイバーにフラッディングし、新しいトポロジ ビューをすぐに収束させます。ネットワーク内の各 OSPF ルータは、すぐに再び同じトポロジ ビューを持ちます。OSPF は、同じ宛先に対する複数の等コストのパスを許容します。すべてのリンクステート情報がフラッディングされて SPF 計算に使用されるため、複数の等コスト パスが計算されてルーティングに使用されることがあります。

ブロードキャスト ネットワークおよび非ブロードキャスト マルチアクセス（NBMA）ネットワークでは、指定ルータ（DR）またはバックアップ DR が LSA フラッディングを実行します。

OSPF は直接 IP の上で実行され、TCP やユーザ データグラム プロトコル（UDP）を使用しません。OSPF はパケット ヘッダーおよび LSA のチェックサムを使用してそれ自体でエラー訂正を実行します。

OSPFv3 は、基本概念は OSPF Version 2 と同じですが、IPv6 の拡大されたアドレス サイズのサポートが追加されています。IPv6 のアドレスとプレフィックスを伝送するために新しい LSA タイプが作成され、個々の IP サブネット ベースではなく、個々のリンク ベースでプロトコルが実行されます。

OSPF は通常多くの内部ルータ間の調整を必要とします。このようなルータには、複数のエリアに接続されたエリア境界ルータ（ABR）や、他のソース（IS-IS、BGP、静的ルートなど）からの再ルーティングを OSPF トポロジに伝達する自律システム境界ルータ（ASBR）があります。OSPF ベースのルータまたはアクセス サーバの最小設定では、すべてのデフォルト パラ

メータ値、およびエリアに割り当てられたインターフェイスが使用され、認証は行われません。環境をカスタマイズする場合は、すべてのルータの調和が取れた設定が必要です。

Cisco IOS XR ソフトウェアの OSPFv3 と OSPFv2 の比較

OSPFv3 プロトコルの大半は OSPFv2 と同じです。OSPFv3 は RFC 2740 に記載されています。

Cisco IOS XR ソフトウェアの OSPFv3 プロトコルと OSPFv2 プロトコルの主な相違点は、次のとおりです。

- OSPFv2 を拡張した OSPFv3 では、IPv6 ルーティング プレフィックスとサイズの大きい IPv6 アドレスのサポートを提供しています。
- NBMA インターフェイスを OSPFv3 で使用する場合、ユーザは、ネイバーのリストを使用してルータを手動で設定する必要があります。隣接ルータはネイバーに接続されたインターフェイスのリンク ローカル アドレスによって識別されます。
- OSPFv2 とは異なり、複数の OSPFv3 プロセスをリンク上で実行できます。
- OSPFv3 の LSA は、「アドレスとマスク」ではなく、「プレフィックスとプレフィックス長」として表現されます。
- ルータ ID は IPv6 アドレスとは無関係な 32 ビットの数値です。

OSPF の階層 CLI および CLI 継承

階層 CLI とは、定義された階層レベル（ルータ レベル、エリア レベル、インターフェイス レベルなど）で、ネットワーク コンポーネント情報がグループ化されたものです。階層 CLI を使うと、OSPF の設定、メンテナンス、トラブルシューティングをより簡単に行えます。コンフィギュレーション コマンドが一緒に階層コンテキストに表示されると、視覚的な検査が簡単になります。階層 CLI はサポートされる CLI 継承自体に備わっています。

CLI 継承を使うと、エリアやインターフェイスのパラメータを明示的に設定する必要がありません。ソフトウェアでは、同じエリアのインターフェイスのパラメータだけを 1 つのコマンドで設定できます。また、エリア コンフィギュレーション レベルやルータ OSPF コンフィギュレーション レベルなどの高い階層レベルからパラメータ値を継承できます。

たとえば、インターフェイスの hello interval 値は、IF ステートメントの優先順位によって次のように決まります。

インターフェイス コンフィギュレーション レベルで hello interval コマンドが設定されている場合は、インターフェイスに設定されている値を使用します。

エリア コンフィギュレーション レベルで hello interval コマンドが設定されている場合は、エリアに設定されている値を使用します。

ルータ コンフィギュレーション レベルで hello interval コマンドが設定されている場合は、ルータ設定されている値を使用します。

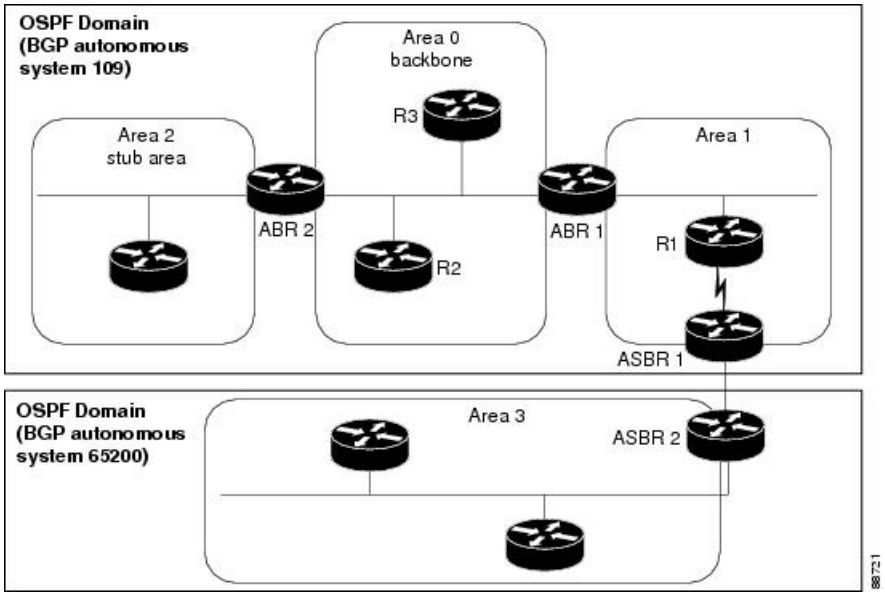
その他の場合は、コマンドのデフォルト値を使用します。

OSPF ルーティング コンポーネント

OSPF を実装する前に、ルーティング コンポーネントの概要とその使用目的を把握する必要があります。これらは自律システム、エリア タイプ、内部ルータ、ABR、および ASBR で構成されます。

図 *4: OSPF* ルーティング コンポーネント

次の図に、OSPF ネットワークのトポロジのルーティング コンポーネントを示します。



自律システム

自律システムは、同じ管理制御下で、相互にルーティング情報を共有するネットワークの集合です。自律システムは、ルーティング ドメインとも呼ばれます。図 *1*：*OSPF* ルーティング コンポーネントには、109 と 65200 の 2 つの自律システムが示されています。 自律システムは 1 つまたは複数の OSPF エリアで構成されます。

エリア

エリアでは、自律システムをより小さく管理しやすいネットワークや隣接ネットワークのセットに再分割できます。図 *1*：*OSPF* ルーティングコンポーネントに示すように、自律システム 109 はエリア 0、エリア 1、エリア 2 の 3 つのエリアから構成されます。

OSPF は 1 つのエリアのトポロジをその他の自律システムから見えないようにします。1 つのエリアのネットワーク トポロジはそのエリア内のルータにのみ認識されます。OSPF ルーティングがエリア内にある場合、そのルーティングはエリア内ルーティングと呼ばれます。このルーティングは、ネットワークにフラッディングするリンク ステート情報量を制限して、ルーティング トラフィックを少なくします。各ルータのトポロジ情報のサイズも小さくし、各ルータの処理と必要なメモリを節約します。

また、エリア内のルータはエリア外の詳細なネットワーク トポロジを見ることはできません。このようにトポロジ情報の開示が制限されているため、自律システム全体が1つのルーティング ドメインであるときに、エリア間のトラフィック フローを制御して、ルーティング トラフィックを少なくすることができます。

バックボーン エリア

バックボーンエリアは、自律システムの複数エリア間でルーティング情報を配布する役割を担当します。エリアの外で発生する OSPF ルーティングをエリア間ルーティングと呼びます。

エリアのプロパティはすべてバックボーン自体にあります。これは、バックボーンだけにあるABR、ルータ、ネットワークで構成されます。図 *1*：*OSPF* ルーティング コンポーネントに示すように、 エリア 0 は OSPF バックボーン エリアです。すべての OSPF バックボーン エリアでは、0.0.0.0 の ID が予約されています。

ルータ

OSPF ネットワークは ABR、ASBR、内部ルータで構成されます。

エリア境界ルータ

エリア境界ルータ（ABR）は複数のエリアのネットワークに直接接続する複数のインターフェイスを持つルータです。ABR は OSPF アルゴリズムのコピーを個別に実行し、バックボーン エリアを含む、アタッチされる各エリアに対する個別のルーティングデータを保持します。また、ABR はアタッチされたエリアの設定の集約をバックボーン エリアに送り、バックボーン エリアではこの情報を自律システム内の他のOSPFエリアに配布します。図 *1*：*OSPF* ルーティング コンポーネント セクションには、2 つの ABR があります。ABR 1 はバックボーン エリアに対するエリア 1 のインターフェイスとなります。ABR 2 はスタブ エリアであるエリア 2 に対するバックボーン エリア 0 のインターフェイスとなります。

自律システム境界ルータ（ASBR）

自律システム境界ルータ（ASBR）を使用すると、1 つの自律システムから別のシステムに接続できるようになります。ASBR は自律システム ルーティング情報を他の自律システムの境界ルータと交換します。自律システム内のすべてのルータは、その自律システムの境界ルータに到達する方法を情報として保有しています。

ASBR は、BGP などの他のプロトコルから外部ルーティング情報をインポートして、それらをネットワークに AS-External（ASE）タイプ 5 LSA として再配布できます。Cisco IOS XR ルータが ASBR の場合、コンテンツの VIP アドレスを自律システムの外部ルートとしてアドバタイズするようにルータを設定できます。このようにして、ASBR は OSPF ネットワーク内のルータに外部ネットワークに関する情報をフラッディングします。

ASBR ルートは、タイプ 1 またはタイプ 2 の ASE としてアドバタイズできます。タイプ 1 とタイプ 2 ではコストの計算方法が異なります。タイプ 2 ASE では、同じ宛先への複数パスを比較するとき、外部コスト（メトリック）のみが考慮されます。タイプ 1 ASE では、外部コストと ASBR に到達するためのコストの組み合わせが使用されます。タイプ 2 の外部コストがデフォルトであり、常に OSPF ルートよりコストがかかるため、OSPF ルートが存在しない場合にのみ使用されます。

内部ルータ

内部ルータ（図 *1*：*OSPF* ルーティング コンポーネントの R1 など）は 1 つの領域に接続されています（たとえば、同一の領域に存在するすべてのインターフェイス）。

OSPF プロセスおよびルータ ID

OSPF プロセスは、物理ルータで OSPF を実行している論理ルーティング エンティティです。システム管理者が物理ボックスをパーティションで個別のルータに区切ることができる論理ルーティング機能がありますが、その機能とこの論理ルーティングエンティティを混同しないでください。

物理ルータは複数の OSPF プロセスを実行できます。ただし、複数のプロセスを実行するのは、複数の OSPF ドメインに接続する場合のみです。各プロセスにはそれぞれのリンクステート データベースがあります。ルーティング テーブルのルートはリンクステート データベースから計算されます。ルートが再配布されないかぎり、1 つの OSPF プロセスは別の OSPF プロセスとルートを共有しません。

各 OSPF プロセスは、ルータ ID で識別されます。ルータ ID はルーティング ドメイン全体で一意である必要があります。OSPF はルータ ID を優先度の高い順に次の送信元から取得します。

- デフォルトでは、OSPF プロセスが初期化されると、チェックポイント データベースに router-id があるかどうかをチェックします。
- ルータ コンフィギュレーション モードで OSPF router-id コマンドで指定された 32 ビット数値。（この値には任意の 32 ビット値を指定できます。このルータのインターフェイスに割り当てられた IPv4 アドレス以外のアドレスを設定できます。また、ルーティング可能な IPv4 アドレスでなくてもかまいません。）
- ITAL が選択した router-id。
- OSPF プロセスが実行されているインターフェイスのプライマリ IPv4 アドレス。OSPF インターフェイスの最初のインターフェイス アドレスが選択されます。

ルータ コンフィギュレーション モードで router-id コマンドを使用してルータ ID を設定することを推奨します。個別の OSPF プロセスは同じルータ ID を共有できますが、その場合、それらのプロセスは同じ OSPF ルーティング ドメインには存在できません。

サポート対象 OSPF ネットワーク タイプ

OSPF は異なるメディアを次のタイプのネットワークに分類します。

- NBMA ネットワーク
- ブロードキャスト ネットワーク

ブロードキャストまたはNBMAネットワークとしてネットワークを設定できます。たとえば、ユーザのネットワークにあるルータでマルチキャスト アドレッシングがサポートされない場合

に、この機能を使用してブロードキャスト ネットワークを NBMA ネットワークとして設定できます。

OSPF のルート認証方法

OSPF Version 2 は 2 種類の認証（プレーン テキスト認証と MD5 認証）をサポートします。デフォルトでは、認証はイネーブルになっていません（RFC 2178 ではヌル認証と呼ばれます）。

OSPV Version 3 では、キー ロールオーバーを除くすべてのタイプの認証がサポートされています。

プレーン テキスト認証

プレーンテキスト認証（タイプ 1 認証とも呼ばれる）では、物理メディアを移動するパスワードを使用します。この認証は、アクセス権限を持ないユーザや、ネットワークに接続するパスワードを使用できないユーザでも簡単に見ることができます。そのため、プレーンテキスト認証はセキュリティで保護されません。プレーンテキスト認証はOSPFインターフェイスの誤った実装や設定ミスにより、間違った OSPF パケットが送信されることを防止できる場合があります。

MD5 認証

MD5 認証はセキュリティで保護されます。パスワードは物理メディアに移動されません。その代わり、ルータでは MD5 を使用して、OSPF パケットとキーのメッセージ ダイジェストが生成され、このメッセージ ダイジェストが物理メディアに送信されます。MD5 認証を使用すると、未認証または悪意のあるルーティング アップデートをルータで受け取らないようにできますが、トラフィックを迂回させることによってネットワーク セキュリティが危険にさらされる可能性があります。

（注） MD5 認証では複数のキーがサポートされています。キー番号をキーに関連付ける必要があります。「*OSPF* 認証のメッセージ ダイジェスト管理」を参照してください。

キー ロールオーバー

OSPF 隣接関係（およびトポロジ）を中断することなく、操作用ネットワークでMD5キーを変更するために、キー ロールオーバー メカニズムがサポートされています。ネットワーク管理者が新しいキーを複数のネットワーキング デバイスに設定するとき、異なるデバイスで新しいキーと古いキーの両方が使用されていることがあります。インターフェイスに新しいキーが設定されている場合、ソフトウェアから 2 つの同じパケットのコピーが送信されます。それぞれのパケットは古いキーと新しいキーによって認証されます。ソフトウェアではどのデバイスが新しいキーの使用を開始したかを追跡し、すべてのネイバーで新しいキーが使用されていることを検出すると、重複パケットの送信を停止します。次に、ソフトウェアでは古いキーを廃棄します。ネットワーク管理者は、各ルータの各コンフィギュレーション ファイルから古いキーを削除する必要があります。

OSPF FIB ダウンロード通知

OSPF FIB ダウンロード通知によって、ライン カードのリロード後の長期間にわたり入力トラフィックのドロップが最小化されます。

Open Shortest Path First（OSPF）は、ITAL を介してルーティング情報ベース（RIB）に登録され、すべてのルートが転送情報ベース（FIB）にダウンロードされるまで、インターフェイスがダウン状態のままになります。OSPF は、リロードされたライン カード上のすべてのルートが RIB/FIB を介してダウンロードされると、インターフェイス アップ通知を取得します。

RIB は、以下の場合に登録クライアントに通知を提供します。

- ノードが失われた。
- ノードが作成された。
- ノードの FIB アップロードが完了した。

OSPF の指定ルータ（DR）

OSPF は、1 つのルータを DR に、もう 1 つのルータを BDR に選択することで、ブロードキャスト セグメントまたは NBMA セグメント上でのみ、セグメント上で交換される情報量を最小化します。このため、セグメント上のルータには、情報交換のための中央接続ポイントがあります。各ルータは、セグメント上の他の各ルータとルーティング アップデートを交換するのではなく、DR および BDR と情報を交換します。DR および BDR は、情報を他のルータに中継します。

ソフトウェアによってセグメント上の各ルータのプライオリティが確認され、DR および BDR となるルータが決定されます。最も高いプライオリティのルータが DR として選択されます。プライオリティが同じ場合、よりの高位ルータ ID を持つルータが優先されます。DR が選択されると、BDR も同様の方法で選択されます。プライオリティが 0 に設定されているルータは、DR または BDR になる資格がありません。

OSPF のデフォルト ルート

タイプ 5（ASE）LSA が生成され、スタブ エリアを除くすべてのエリアにフラッディングされます。スタブ エリアにあるルータから、スタブ エリア外の宛先にパケットをルーティングできるようにするために、スタブ エリアにアタッチされている ABR によってデフォルト ルートが挿入されます。

デフォルト ルートのコストは 1 です（デフォルト）。または、default-cost コマンドに指定されている値によって決まります。

OSPF Version 2 のリンクステート アドバタイズメント タイプ

次の各 LSA タイプには、個別の目的があります。

- ルータ LSA（タイプ 1）：1 つのエリア内にルータが持つリンクと各リンクのコストを表します。これらのLSAは、エリア内でのみフラッディングされます。LSAは、QoS（Quality of Service）に基づいてルータがパスを計算できるかどうか、ルータが ABR または ASBR のどちらであるか、ルータが仮想リンクの一端であるかどうかを示します。また、タイプ 1 の LSA は、スタブ ネットワークへのアドバタイズにも使用されます。
- ネットワーク LSA（タイプ 2）：マルチアクセス ネットワーク セグメントにアタッチされているすべてのルータに関するリンク ステートとコストの情報を表します。この LSA ではネットワーク セグメントにアタッチされているインターフェイスを持つすべてのルータを一覧にします。この LSA のコンテンツを生成して追跡するのは、ネットワーク セグメントの指定ルータの仕事です。
- ABR のサマリーLSA（タイプ3）：他のエリア内のルータ（エリア間ルート）に内部ネットワークをアドバタイズします。タイプ 3 の LSA は、1 つのネットワークを表すことも、1 つのプレフィックスに集約された一連のネットワークを表すこともあります。サマリー LSA を生成するのは ABR だけです。
- ASBR のサマリー LSA（タイプ 4）：ASBR および ASBR に到達するまでのコストをアドバタイズします。外部ネットワークにアクセスしようとするルータは、これらのアドバタイズメントを使用して、ネクストホップへの最適パスを決定します。ABR はタイプ 4 LSA を生成します。
- 自律システム外部 LSA（タイプ 5）：別の自律システムからルートを再配布します。通常は別のルーティング プロトコルから OSPF に再配布します。
- 自律システム外部 LSA（タイプ 7）：外部ルート情報を NSSA 内で伝搬するために提供されます。タイプ 7 LSA は NSSA で生成およびアドバタイズできます。NSSA はタイプ 5 LSA を受信または生成しません。タイプ 7 LSA は 1 つの NSSA 内でのみアドバタイズされます。境界ルータによってバックボーンエリアや他のエリアにフラッディングされることはありません。
- 内部エリア プレフィックス LSA（タイプ 9）：ルータは各ルータまたは中継ネットワークに複数の内部エリア プレフィックス LSA を生成できます。それぞれの内部エリア プレフィックス LSA には固有のリンクステート ID があります。それぞれの内部エリア プレフィックス LSA のリンクステート ID には、ルータ LSA またはネットワーク LSA に対する関係と、スタブおよび中継ネットワークのプレフィックスが記されています。
- エリア ローカル スコープ（タイプ 10）：Opaque LSA は関連付けられているエリアの境界を越えてフラッディングされません。
- リンクステート（タイプ 11）：LSA は AS を通してフラッディングされます。タイプ 11 LSA のフラッディング スコープは、AS-External（タイプ 5）LSA のフラッディング スコープと同じです。タイプ 5 LSA と同様、タイプ 11 Opaque LSA がスタブ エリア内の隣接ルータからスタブ エリアに受信されると、LSA は拒否されます。タイプ 11 Opaque LSA には、次のような属性があります。
  - LSA はすべての中継エリアを超えてフラッディングされます。
  - LSA はバックボーンからのスタブ エリアにはフラッディングされません

• LSA はルータから、ルータが接続されたスタブ エリアには発信されません。

OSPFv3 のリンクステート アドバタイズメント タイプ

次の各 LSA タイプには、個別の目的があります。

- ルータ LSA（タイプ 1）：リンク ステートおよびエリアに対するルータ リンクのコストを表します。これらの LSA は、エリア内でのみフラッディングされます。LSA は、ルータが ABR または ASBR のどちらであるか、および仮想リンクの一端であるかどうかを示します。また、タイプ 1 の LSA は、スタブ ネットワークへのアドバタイズにも使用されます。OSPFv3 では、これらの LSA はアドレス情報を持たず、ネットワーク プロトコルに依存しません。OSPFv3 では、ルータ インターフェイス情報は複数のルータ LSA 間で拡散されます。受信者は、SPF 計算を実行する前に、特定のルータから発信されたすべてのルータ LSA を連結する必要があります。
- ネットワーク LSA（タイプ 2）：マルチアクセス ネットワーク セグメントにアタッチされているすべてのルータに関するリンク ステートとコストの情報を表します。この LSA ではネットワーク セグメントにアタッチされているインターフェイスを持つすべての OSPF ルータを一覧にします。ネットワーク セグメントに選択された指定ルータだけが、セグメントのネットワーク LSA を生成して追跡できます。OSPFv3 では、ネットワーク LSA はアドレス情報を持たず、ネットワーク プロトコルに依存しません。
- ABR のエリア間プレフィックス LSA（タイプ 3）：他のエリア内のルータ（エリア間ルート）に内部ネットワークがアドバタイズされます。タイプ 3 の LSA は、1 つのネットワークを表すことも、1 つのプレフィックスとして集約された一連のネットワークを表すこともあります。ABR はタイプ 3 LSA だけを生成します。OSPFv3 では、これらの LSA のアドレスは「address および mask」ではなく「prefix および prefix length」で表されます。デフォルト ルートは、長さが 0 のプレフィックスとして表現されます。
- ASBR のエリア間ルータ LSA（タイプ 4）：ASBR および ASBR に到達するまでのコストをアドバタイズします。外部ネットワークにアクセスしようとするルータは、これらのアドバタイズメントを使用して、ネクスト ホップへの最適パスを決定します。ABR はタイプ 4 LSA を生成します。
- 自律システム外部 LSA（タイプ 5）：別の自律システムからルートを再配布します。通常は別のルーティング プロトコルから OSPF に再配布します。OSPFv3 では、これらの LSA のアドレスは「address および mask」ではなく「prefix および prefix length」で表されます。デフォルト ルートは、長さが 0 のプレフィックスとして表現されます。
- 自律システム外部 LSA（タイプ 7）：外部ルート情報を NSSA 内で伝搬するために提供されます。タイプ 7 LSA は NSSA で生成およびアドバタイズできます。NSSA はタイプ 5 LSA を受信または生成しません。タイプ 7 LSA は 1 つの NSSA 内でのみアドバタイズされます。境界ルータによってバックボーン エリアや他のエリアにフラッディングされることはありません。
- リンク LSA（タイプ 8）：リンクローカル フラッディング スコープを持ち、関連付けられているリンクを超えてフラッディングすることはありません。リンク LSA は、リンク

またはネットワーク セグメントに接続されている他のすべてのルータに対してルータのリンクローカルアドレスを提供し、リンクに接続されている他のルータに、そのリンクに関連付ける IPv6 プレフィックスのリストを通知します。また、ルータが Options ビットの集まりをアサートして、リンクの起点となるネットワーク LSA と関連付けできるようにします。

- 内部エリア プレフィックス LSA（タイプ 9）：ルータは各ルータまたは中継ネットワークに複数の内部エリア プレフィックス LSA を生成できます。それぞれの内部エリア プレフィックス LSA には固有のリンクステート ID があります。それぞれの内部エリア プレフィックス LSA のリンクステート ID には、ルータ LSA またはネットワーク LSA に対する関係と、スタブおよび中継ネットワークのプレフィックスが記されています。

新しく定義された LSA のほとんどすべてに、アドレス プレフィックスが存在します。プレフィックスは、Prefix Length、Prefix Options、および Address Prefix の 3 つのフィールドで表現されます。OSPFv3 では、これらの LSA のアドレスは「address および mask」ではなく「prefix および prefix length」で表されます。デフォルト ルートは、長さが 0 のプレフィックスとして表現されます。

エリア間プレフィックス LSA およびエリア内プレフィックス LSA では、すべての IPv6 プレフィックス情報が伝送されます。IPv4 ではこの情報はルータ LSA およびネットワーク LSA に含まれます。特定の LSA（ルータ LSA、ネットワーク LSA、エリア間ルータ LSA、およびリンク LSA）の Options フィールドは、IPv6 で OSPF をサポートするために 24 ビットに拡張されています。

OSPFv3 では、エリア間プレフィックス LSA、エリア間ルータ LSA、および自律システム外部 LSA のリンクステート ID の機能は、リンクステート データベースの個々の部分を識別することだけです。OSPF Version 2 ではリンクステート ID で表されたアドレスまたはルータ ID はすべて、OSPFv3 では LSA の本体で伝送されます。

パッシブ インターフェイス

パッシブとしてインターフェイスを設定すると、ネイバーへのルーティング アップデートの送信が無効になるため、隣接関係は OSPF で形成されません。ただし、特定のサブネットは OSPF ネイバーに引き続きアドバタイズされます。インターフェイスでの OSPF プロトコル動作の送信を抑制するには、適切なモードで **passive** コマンドを使用します。

ホストを持つLANセグメントをネットワークの残りに接続しているが、ルータ間のトランジットリンクになるように作られていないインターフェイスでは、パッシブ設定を使用することを推奨します。

グレースフル リスタート操作のモード

ルータがこの機能に使用できる動作モードは、再起動モードとヘルパー モード再起動モード、ヘルパー モード、およびプロトコル シャットダウン モードです。再起動モードは、OSPFv3 プロセスがグレースフル リスタートを実行しているときに開始されます。ヘルパー モードでは、OSPFv3 が隣接ルータで再起動している間に、確立されている OSPFv3 ルートでトラフィックを転送し続けている隣接ルータを参照します。

リスタート モード

OSPFv3 プロセスが開始されたときに、グレースフル リスタートを試行する必要があるかどうかを決定します。決定は、グレースフル リスタートがそれまでにイネーブルされているかどうかに基づきます。（OSPFv3 は、ルータの初回の起動時にグレースフル リスタートを試行しません。）OSPFv3 グレースフル リスタートを有効にすると、RIB の消去タイマーがゼロ以外の値に変わります。

グレースフル リスタート中、ルータは OSPFv3 ルートを RIB に入力しません。ルータは再起動前に OSPFv3 が保有していた完全に隣接するネイバーとの完全な隣接関係を立ち上げようとします。最終的に、OSPFv3 プロセスは、（何らかの理由により）グレースフル リスタートを終了するため、または、グレースフル リスタートを終了したため、プロセスがコンバージされたことを RIB に示します。

最後の再起動から間を空けずに OSPFv3 が再起動を試みると、OSPFv3 プロセスは頻繁に繰り返しクラッシュするようになり、新しいグレースフルリスタートの実行が停止します。グレースフルリスタートの許可間隔を制御するには、graceful-restart interval コマンドを使用します。起動する最初のインターフェイスで OSPFv3 がグレースフル リスタートを開始すると、グレースフル リスタートの期間（有効期間）を制限するためにタイマーが起動します。graceful-restart lifetime コマンドを使用して、この期間を設定できます。起動する各インターフェイスで *grace* LSA（タイプ 11）がフラッディングされ、このルータがグレースフル リスタートを試みていることを隣接ルータに示します。ネイバーはヘルパー モードを開始します。再起動中のネイバーから受信した hello パケットの指定ルータとバックアップ指定ルータ パケットの指定ルータ チェックは正しくないため、バイパスされます。

ヘルパー モード

ヘルパー モードは、デフォルトでイネーブルになっています。グレースフル リスタートを試みているルータから（ヘルパー）ルータが grace LSA（タイプ 11）を受け取ると、次のイベントが発生します。

- graceful-restart helper disable コマンドによりヘルパー モードがディセーブルされている場合、ルータは LSA パケットをドロップします。
- ヘルパー モードがイネーブルの場合、次の条件がすべて満たされると、ルータはヘルパー モードを開始します。
  - ローカル ルータ自体がグレースフル リスタートを試みていない。
  - ローカル（ヘルパー）ルータに送信先ネイバーとの完全な隣接関係がある。
  - 受信した LSA の *lsage*（リンク ステートの経過時間）の値が、要求された猶予期間よりも短い。
  - grace LSA の送信元が grace LSA の生成元と同じである。
- ヘルパー モードを開始すると、ルータは一定期間そのヘルパー機能を実行します。この期間は再起動モードにあるルータの有効期間の値から、受信した grace LSA の *lsage* の値を引いた値です。グレースフル リスタートが時間内に成功すると、ヘルパー タイマーが期

限切れになる前に停止します。ヘルパー タイマーの期限が切れた場合、再起動しているルータへの隣接関係がダウンし、通常の OSPFv3 機能が再開します。

- デッド タイマーはヘルパー モードにあるルータでは使用できません。
- 次のいずれかの場合に、ヘルパー モードにあるルータはヘルパー機能の実行を停止します。
  - ヘルパー ルータが再起動中のルータとの完全な隣接関係を起動できる。
  - ヘルパー機能のローカル タイマーの有効期限が切れている。

プロトコル シャットダウン モード

このモードでは、OSPFv3 操作は完全に無効になっています。これは、自己生成リンク ステート アドバタイズメント（LSA）をフラッシュすることで達成され、ローカルの OSPFv3 対応インターフェイスが即座に停止し、リンク ステートデータベース（LSDB）がクリアされます。ローカル以外の LSDB エントリは OSPFv3 によって削除され、フラッディング（MaxAged）されません。

プロトコル シャットダウン モードは、**protocol shutdown** コマンドを使用（プロトコル インスタンスが無効になります）して手動で起動できます。または OSPFv3 プロセスのメモリが不足すると起動します。次のイベントは、プロトコル シャットダウンが実行されると発生します。

- ローカル ルータ LSA およびすべてのローカル リンク LSA がフラッシュされます。他の LSA はすべて、ドメイン内の他の OSPFv3 ルータによって最終的にエージ アウトされます。
- まだローカル ルータとフル状態になっていない OSPFv3 ネイバーは、Kill_Nbr イベントとともに停止します。
- 3 秒の遅延後、空の Hello パケットはアクティブな隣接関係がある各ネイバーに即座に送信されます。
  - 空の Hello パケットは、dead_interval が経過するまで定期的に送信されます。
  - dead_interval が経過すると、Hello パケットは送信されなくなります。

Dead Hello インターバルの遅延（4 X Hello インターバル）後、次のイベントが実行されます。

- その OSPFv3 インスタンスからの LSA データベースがクリアされます。
- OSPFv3 によってインストールされた RIB からのすべてのルートが消去されます。

ルータは、プロトコル シャットダウン状態時にネイバーから受信するいずれの OSPF 制御パケットにも応答しません。

プロトコルの復元

プロトコルを復元する方法は、シャットダウンを最初に引き起こしたトリガーに依存します。OSPFv3 が **protocol shutdown** コマンドを使用してシャットダウンされた場合、OSPFv3 を通常

の動作に復元するには **no protocol shutdown** コマンドを使用します。OSPFv3 が sysmon からの重要なメモリ メッセージによってシャットダウンされた場合は、十分なメモリがプロセッサに復元されたことを示す sysmon からの通常のメモリ メッセージによって OSPFv3 プロトコルが復元され、通常の動作が再開されます。OSPFv3 が重要なメモリ トリガーによってシャットダウンされた場合は、通常のメモリ レベルがルート プロセッサで復元された際に、手動で再起動する必要があります。これは自動的に復元されません。

次のイベントは、OSPFv3 が復元されると発生します。

1. すべての OSPFv3 インターフェイスが、Hello パケットとデータベース交換を使用してバックアップされます。
2. ローカル ルータおよびリンク LSA が再作成され、アドバタイズされます。
3. ルータは、ネイバーから受信したすべての OSPFv3 制御メッセージに正常に応答します。
4. 他の OSPFv3 ルータから学習されたルートが RIB にインストールされます。

OSPF Version 2 および OSPFv3 でのロード バランシング

ルータは、複数のルーティング プロセス（またはルーティング プロトコル）を使用して特定のネットワークへの複数のルートを認識すると、最短のアドミニストレーティブ ディスタンスを持つルートをルーティング テーブルにインストールします。同じアドミニストレーティブ ディスタンスを持つ同じルーティング プロセスを使用して認識された多数のルートから、1 つのルートを選択する必要があることもあります。この場合、ルータはその宛先へのコスト（またはメトリック）が最も小さいパスを選択します。各ルーティング プロセスはコストをそれぞれの方法で計算します。コストは、ロードバランシングを実現するために処理が必要なこともあります。

OSPF では、自動的にロードバランシングが実行されます。OSPF により、複数のインターフェイスを通って宛先に到達できること、および各パスのコストが同じであることが検出された場合は、ルーティング テーブルに各パスがインストールされます。同じ宛先へのパスの数は、**maximum-paths** （OSPF） コマンドを指定しない限り、制限されません。

最大パスの範囲は 1 から 8 です。デフォルトの最大パスの数は 8 です。

OSPFv2 のパス計算要素

PCE はネットワーク パスやルートをネットワーク図に基づいて計算し、計算上の制限を適用する機能を持つエンティティ（コンポーネント、アプリケーション、ネットワーク ノード）です。

PCE は、PCE アドレスおよびクライアントが MPLS-TE に設定されると実行されます。PCE はその PCE アドレスおよび機能を OSPF に通信して、OSPF はこの情報を PCE ディスカバリ Type-Length-Value（TLV）（タイプ 2）にパッケージ化し、RI LSA を再発信します。OSPF には、すべての RI LSA でルータ機能 TLV（タイプ 1）も含まれます。PCE ディスカバリ TLV には PCE アドレス サブ TLV（タイプ 1）およびパス スコープ サブ TLV（タイプ 2）が含まれます。

PCE アドレス サブ TLV では PCE に到達するために使用される必要がある IP アドレスを指定します。このアドレスは常に到達可能なループ バック アドレスにする必要があります。この TLV は必須であり、PCE ディスカバリ TLV 内に存在する必要があります。パス スコープ サブ TLV は、PCE パス計算スコープを示します。これは、PCE 機能を参照して計算したり、エリア内ルート、エリア間、AS 間、またはレイヤ TE 間 LSP の計算に参加したりします。

OSPFv2 への PCE 拡張機能には、ルータ情報リンク ステート アドバタイズメント（RI LSA）のサポートが含まれます。OSPFv2 は、すべてのエリアの範囲（LSA タイプ 9、10、および 11）を受信するように拡張されます。ただし、OSPFv2 はエリア範囲タイプ 10 のみを発信します。

パス計算要素機能の詳細については、『*MPLS Configuration guide*』の「*Implementing MPLS Traffic Engineering*」モジュール、および次の IETF ドラフトを参照してください。

- draft-ietf-ospf-cap-09
- draft-ietf-pce-disco-proto-ospf-00

OSPFv3 の管理情報ベース（MIB）

Cisco IOS XR では RFC 5643 に定義されている MIB および OSPFv3 のトラップが完全にサポートされています。RFC 5643 には、IPv6 用の Open Shortest Path First（OSPF）ルーティング プロトコル（OSPF バージョン 3）で使用する管理情報ベース（MIB）のオブジェクトが定義されています。

OSPFv3 MIB の実装は、IETF ドラフト『*Management Information Base for OSPFv3 (draft-ietf-ospf-ospfv3-mib-8)*』に基づきます。RFC 5643 にアップグレードすると、ユーザは新しい MIB をピックアップするように NMS アプリケーションを更新する必要があります。

複数の OSPFv3 インスタンス

SNMPv3 は、複数の OSPFv3 インスタンスに MIB ビューを設定するために使用できる「コンテキスト」を同じシステムでサポートします。

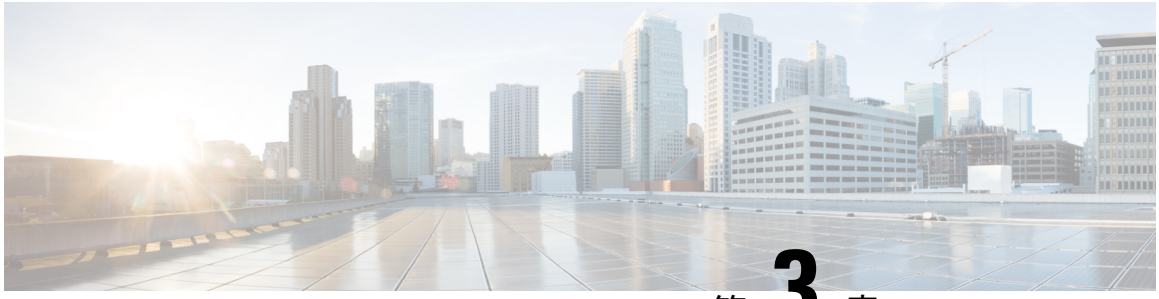
OSPFv2 の VRF-lite サポート

OSPF バージョン 2（OSPFv2）の VRF Lite 機能は、イネーブルになっています。VRF-Lite は、BGP/MPLS ベースのバックボーンがない状態での仮想ルーティングおよび転送（VRF）導入です。VRF-Lite では、個別のプロバイダー エッジ（PE）ルータは VRF インターフェイスを使用して直接接続されています。OSPFv2 の VRF-Lite をイネーブルにするには、VRF コンフィギュレーション モードで **capability vrf-lite** コマンドを設定します。VRF-Lite が設定されている場合、DN ビット処理および自動エリア境界ルータ（ABR）のステータス設定はディセーブルです。

OSPFv3 タイマーの更新

Open Shortest Path First バージョン 3（OSPFv3）タイマー リンク ステート アドバタイズメント（LSA）、Shortest Path First（SPF）スロットリングのデフォルト値は次のように更新されます。

- **timers throttle lsa all**：*start-interval*：50 ミリ秒および *hold-interval*：200 ミリ秒
- **timers throttle spf** ：*spf-start*：50 ミリ秒、*spf-hold*：200 ミリ秒、*spf-max-wait*：5000 ミリ秒



第 3 章

RIB の実装とモニタリング

ルーティング情報ベース（RIB）は、ネットワークのすべてのノード間のルーティングの接続に関する情報を収集して配布したものです。各ルータには、そのルータのルーティング情報を含む RIB を維持します。RIB は、システムで実行されているすべてのルーティング プロトコルでの最良ルートを保存します。

各ルーティングプロトコルは独自の最適ルートのセットを選択し、これらのルートとその属性を RIB に取り込みます。RIB はこれらのルートを格納し、すべてのルーティング プロトコルの中から最適ルートを選択します。これらのルートは転送パケットで使用するために、ラインカードにダウンロードされます。頭字語の RIB は、RIB プロセスと、RIB 内に含まれるルートデータの集合を表すために使用されます。プロトコル内で、ルートはそのプロトコルによって使用されているメトリックに基づいて選択されます。プロトコルは最適なルート（最も低いメトリックまたは結び付けられたメトリック）を RIB にダウンロードします。RIB は、関連付けられているプロトコルのアドミニストレーティブ ディスタンスを比較して、全体的に最適なルートを選択します。

このモジュールでは、ネットワークで RIB を実装およびモニタリングする方法を説明します。

（注） VPNv4、VPNv6 および VPN ルーティング/転送（VRF）のアドレス ファミリは、今後のリリースでサポートされる予定です。

- ルーティング テーブルを使用した RIB 設定の確認 （116 ページ）
- ネットワーキングとルーティングの問題の検証 （117 ページ）
- RIB ネクストホップ ダンプニングのディセーブル化 （118 ページ）
- RCC および LCC オンデマンド スキャンのイネーブル化 （119 ページ）
- RCC および LCC バックグラウンド スキャンのイネーブル化 （120 ページ）
- RIB の参照 （122 ページ）

ルーティング テーブルを使用した RIB 設定の確認

ルーティング テーブルの概要と詳細の情報をチェックすることで、RIB が RP 上で実行され、正常に機能していることを確認するために、RIBの設定を確認するには、次の作業を実行します。

手順

ステップ 1 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **summary** [**detail**] [**standby**]

例：

```
RP/0/RP0/cpu 0: router# show route summary
```

指定したルーティング テーブルに関するルート サマリー情報を表示します。

- 要約されたデフォルト テーブルは、IPv4 ユニキャスト ルーティング テーブルです。

ステップ 2 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] [*protocol* [*instance*] | *ip-address mask*] [**standby**] [**detail**]

例：

```
RP/0/RP0/cpu 0: router# show route ipv4 unicast
```

指定したルーティング テーブルに関する詳細なルート情報を表示します。

- このコマンドは、表示を制限するために通常は IP アドレスまたは他のオプション フィルタを使用して発行します。それ以外の場合は、デフォルトの IPv4 ユニキャスト ルーティングテーブルからすべてのルートを表示します。ネットワークの設定に応じて大規模なリストになる可能性があります。

show route best-local コマンドの出力：例

次に、**show route backup** コマンドの出力例を示します。

```
show route backup

 Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
        O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local
 S    172.73.51.0/24 is directly connected, 2d20h, HundredGigE 0/9/0/0
```

```
Backup  O E2 [110/1] via 10.12.12.2, HundredGigE 0/9/0/0
```

ネットワーキングとルーティングの問題の検証

ノード間のルートの動作を検証するには、次のタスクを実行します。

手順

ステップ 1 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] [*protocol* [*instance*] | *ip-address mask*] [**standby**] [**detail**]

例：

```
RP/0/RP0/cpu 0: router# show route ipv4 unicast 192.168.1.11/8
```

RIB の現在のルートを表示します。

ステップ 2 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **backup** [*ip-address*] [**standby**]

例：

```
RP/0/RP0/cpu 0: router# show route ipv4 unicast backup 192.168.1.11/8
```

RIB のバックアップ ルートを表示します。

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

例：

```
RP/0/RP0/cpu 0: router# show route ipv4 unicast best-local 192.168.1.11/8
```

特定の宛先からの応答パケットに使用する場合に最善のローカル アドレスが表示されます。

ステップ 4 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **connected** [**standby**]

例：

```
RP/0/RP0/cpu 0: router# show route ipv4 unicast connected
```

ルーティング テーブルの現在の接続ルートを表示します。

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

例：

```
RP/0/RP0/cpu 0: router# show route ipv4 unicast local
```

ルーティング テーブルの受信エントリのローカル ルートを表示します。

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

例：

```
RP/0/RP0/cpu 0: router# show route ipv4 unicast longer-prefixes 192.168.1.11/8
```

指定のネットワークと指定の数のビットを共有する RIB の現在のルートを表示します。

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

例：

```
RP/0/RP0/cpu 0: router# show route ipv4 unicast next-hop 192.168.1.34
```

宛先アドレスまでのネクスト ホップ ゲートウェイまたはホストを表示します。

---

show route コマンドの出力：例

次に、アドレスを指定せずに入力した show route コマンドの出力例を示します。

```
show route

  Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
         O - OSPF, IA - OSPF inter area
         N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
         E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
         i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
         ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
         U - per-user static route, o - ODR, L - local

  Gateway of last resort is 172.23.54.1 to network 0.0.0.0

  C    10.2.210.0/24 is directly connected, 1d21h, Ethernet0/1/0/0
  L    10.2.210.221/32 is directly connected, 1d21h, Ethernet0/1/1/0
  C    172.20.16.0/24 is directly connected, 1d21h, GigabitEthernet 0/4/0/0
  L    172.20.16.1/32 is directly connected, 1d21h, GigabitEthernet 0/4/0/0
  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
```

RIB ネクストホップ ダンプニングのディセーブル化

RIB ネクスト ホップ ダンプニングをディセーブルにするには、次のタスクを実行します。

手順

ステップ 1 **router rib**

例：

```
RP/0/RP0/cpu 0: router# route rib
```

RIB コンフィギュレーション モードを開始します。

ステップ 2 **address-family** { **ipv4** | **ipv6** } **next-hop dampening disable**

例：

```
RP/0/RP0/cpu 0: router(config-rib)# address family ipv4 next-hop dampening disable
```

IPv4 アドレス ファミリのネクスト ホップ ダンプニングをディセーブルにします。

ステップ 3 **commit**

show route next-hop コマンドの出力：例

次に、**show route resolving-next-hop** コマンドの出力例を示します。

```
show route resolving-next-hop 10.0.0.1

Nexthop matches 0.0.0.0/0
  Known via "static", distance 200, metric 0, candidate default path
  Installed Aug 18 00:59:04.448
  Directly connected nexthops

    172.29.52.1, via MgmtEth0/RP1/CPU0/0
      Route metric is 0
```

RCC および LCC オンデマンド スキャンのイネーブル化

ルート整合性チェッカ（RCC）、およびラベル整合性チェッカ（LCC）オンデマンドスキャンをトリガーするには、次の作業を実行します。オンデマンドスキャンは、特定のアドレスファミリ（AFI）で、サブアドレスファミリ（SAFI）、テーブル、および、プレフィックス、VRF、またはテーブルのすべてのプレフィックスに関して実行できます。

手順

ステップ 1 次のいずれかのコマンドを使用します。

- **show rcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*]
- **show lcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*]

例：

```
RP/0/RP0/cpu 0: router#show rcc ipv6 unicast 2001:DB8::/32 vrf vrf_1
```

または

```
RP/0/RP0/cpu 0: router#show lcc ipv6 unicast 2001:DB8::/32 vrf vrf_1
```

ルート整合性チェッカ（RCC）またはラベル整合性チェッカ（LCC）オンデマンドで実行します。

ステップ 2 次のいずれかのコマンドを使用します。

- **clear rcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*] **log**
- **clear lcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*] **log**

例：

```
RP/0/RP0/cpu 0: router#clear rcc ipv6 unicast log
```

または

```
RP/0/RP0/cpu 0: router#show lcc ipv6 unicast log
```

以前のスキャンのログをクリアします。

RCC および LCC バックグラウンド スキャンのイネーブル化

ルート整合性チェッカ（RCC）およびラベル整合性チェッカ（LCC）のバックグラウンド スキャンを実行するには、次のタスクを実行します。

手順

ステップ 1 **configure**

ステップ 2 次のいずれかのコマンドを使用します。

- **rcc** {**ipv4** | **ipv6**} **unicast** {**enable** | **period** *milliseconds*}
- **lcc** {**ipv4** | **ipv6**} **unicast** {**enable** | **period** *milliseconds*}

例：

```
RP/0/RP0/cpu 0: router(config)#rcc ipv6 unicast enable
```

```
RP/0/RP0/cpu 0: router(config)#rcc ipv6 unicast period 500
```

または

```
RP/0/RP0/cpu 0: router(config)#lcc ipv6 unicast enable

RP/0/RP0/cpu 0: router(config)#lcc ipv6 unicast period 500
```

RCC または LCC バックグラウンド スキャンをトリガーします。検証のトリガー頻度を制御するには、**period** オプションを使用します。スキャンをトリガーするたびに、転送情報ベース（FIB）に送信されたルートまたはラベルの残りの 1 バッファ分の場所から検証が再開されます。

ステップ 3 **commit**

ステップ 4 次のいずれかのコマンドを使用します。

- **show rcc** {**ipv4**| **ipv6**} **unicast** [**summary** | **scan-id** *scan-id-value*]
- **show lcc** {**ipv4**| **ipv6**} **unicast** [**summary** | **scan-id** *scan-id-value*]

例：

```
RP/0/RP0/cpu 0: router#show rcc ipv6 unicast statistics scan-id 120
```

または

```
RP/0/RP0/cpu 0: router#show lcc ipv6 unicast statistics scan-id 120
```

バックグラウンド スキャンに関する統計情報を表示します。

- **summary**：現在進行中のスキャン ID および以前の少数のスキャンの要約を表示します。
- **scan-id** *scan-id-value*：特定のスキャンに関する詳細情報を表示します。

RCC および LCC のイネーブル化：例

次に、ルート整合性チェッカ（RCC）バックグラウンドスキャンを IPv6 ユニキャスト テーブルのスキャンのバッファ間 500 ミリ秒の時間でイネーブルにする例を示します。

```
rcc ipv6 unicast period 500
```

次に、ラベル整合性チェッカ（LCC）バックグラウンドスキャンを IPv6 ユニキャスト テーブルのスキャンのバッファ間 500 ミリ秒の時間でイネーブルにする例を示します。

```
lcc ipv6 unicast period 500
```

次に、vrf1 のサブネット 10.10.0.0/16 のルート整合性チェッカ（RCC）オンデマンド スキャンを行う例を示します。

```
show rcc ipv4 unicast 10.10.0.0/16 vrf vrf 1
```

次に、ラベル整合性チェッカ（LCC）オンデマンドスキャンを IPv6 プレフィックスのすべてのラベルで実行する例を示します。

```
show lcc ipv6 unicast all
```

RIB の参照

この項では、RIBに関する追加の概念情報について説明します。説明する項目は次のとおりです。

• BGP およびその他のプロトコルでの RIB データ構造 （122 ページ）
• RIB アドミニストレーティブ ディスタンス （122 ページ）
• RIB 統計情報 （123 ページ）
• RIB 隔離 （124 ページ）
• ルートとラベルの整合性チェッカ （124 ページ）

BGP およびその他のプロトコルでの RIB データ構造

RIB は、ボーダー ゲートウェイ プロトコル（BGP）や他のユニキャスト ルーティング プロトコルなどの他のルーティングアプリケーションとは異なるプロセスを使用してデータ構造を維持します。ただし、これらのルーティングプロトコルは、RIBが使用するものと似た内部データ構造を使用し、RIBとしてそのデータ構造を内部的に参照することがあります。たとえば、BGP ルートは BGP RIB（BRIB）に保存されます。RIB プロセスは、BGP によって処理される BRIB に関与しません。

パケットを転送するためにラインカードおよび RP によって使用されるテーブルは、転送情報ベース（FIB）と呼ばれます。RIB プロセスは FIB を構築しません。代わりに、RIB はバルクコンテンツ ダウンローダ（BCDL）プロセスによって、FIBプロセスに最適な、選択されたルートのセットをバルク各ラインカードにダウンロードします。続いて、FIB が構築されます。

RIB アドミニストレーティブ ディスタンス

転送は最長プレフィックス照合に基づいて行われます。10.0.2.1 宛てのパケットを転送する場合、マスク /24 は /16 よりも長い（より具体的である）ため、10.0.2.0/24 は 10.0.0.0/16 よりも優先されます。同じプレフィックスと同じ長さを持つ、異なるプロトコルからのルートは、アドミニストレーティブ ディスタンスに基づいて選択されます。たとえば、Open Shortest Path First（OSPF）プロトコルのアドミニストレーティブ ディスタンスは 110、Intermediate System-to-Intermediate System（IS-IS）プロトコルのアドミニストレーティブ ディスタンスは 115 です。IS-IS および OSPF の両方が RIB に 10.0.1.0/24 をダウンロードすると、OSPF のアドミニストレーティブ ディスタンスの方が小さいため、RIB は OSPF ルートを優先します。同じ長さの複数のルート間で選択するためだけにアドミニストレーティブディスタンスが使用されます。

次の表に、一般的なプロトコルのデフォルトのアドミニストレーティブディスタンスを示します。

表 *1:* デフォルトのアドミニストレーティブ ディスタンス

プロトコル	アドミニストレーティブディスタンスのデフォルト
接続されているルートまたはローカルルート	0
スタティック ルート	1
外部 BGP ルート	20
OSPF ルート	110
IS-IS ルート	115
内部 BGP ルート	200

一部のルーティング プロトコル（たとえば、IS-IS、OSPF、BGP など）のアドミニストレーティブ ディスタンスは変更できます。プロトコルのアドミニストレーティブ ディスタンスを変更する適切な方法については、そのプロトコル固有のマニュアルを参照してください。

（注） すべてではなく一部のルータで、プロトコルのアドミニストレーティブディスタンスを変更すると、ルーティングループなどの予想外の動作が発生することがあります。したがって、これは推奨されません。

RIB 統計情報

RIB は、RIB とクライアントとの間でやり取りされるメッセージ（要求）の統計情報をサポートします。プロトコル クライアントは、メッセージを RIB に送信します（たとえば、ルート追加、ルート削除、ネクスト ホップの登録など）。RIB もメッセージを送信します（たとえば、ルート、アドバタイズメント、ネクスト ホップ通知などの再配布）。これらの統計情報は、どのようなメッセージが送信されたかに関して、また送信されたメッセージ数に関する情報を収集するために使用されます。これらの統計情報には、RIBサーバとそのクライアント間で転送される各種メッセージのカウンタが含まれています。統計情報は、show rib statistics コマンドを使用して表示します。

RIBは、次に挙げるような、クライアントから送信されるすべての要求のカウンタを保持します。

- ルートの動作
- テーブルの登録
- ネクストホップの登録

- 再配布の登録
- 属性の登録
- 同期の完了

RIB は、RIB によって送信されるすべての要求のカウンタも保持します。設定は RIB ネクストホップ ダンプニング機能をディセーブルにします。この結果、クライアントが登録したネクスト ホップが解決された、または解決されなかった場合に RIB はクライアントにすぐに通知します。RIB は、要求の結果に関する情報も保持します。

RIB 隔離

RIB 検疫は、ルーティング プロトコルと RIB 間の相互作用における問題を解決します。問題は、ルートが継続的に挿入され、RIB から取り消される場合に発生する、RIB とルーティング プロトコルの間の持続振動です。問題が解決されるまで、CPU使用率にスパイクが生じます。振動に減衰がない場合、プロトコル プロセスおよび RIB プロセスが CPU を多く使用するため、システムのその他の部分に影響を与え、さらにプロトコルおよびRIBのその他の動作の障害となります。この問題は、RIBにルートの特定の組み合わせが受信されて取り込まれた場合に発生します。この問題は、通常、ネットワークの設定が間違っている場合に発生します。ただし、設定ミスはネットワーク全体であるため、単一のルータの設定時に問題を検出できません。

隔離メカニズムでは相互に再帰的なルートが検出されますが、ここで隔離されるのは相互の再帰が完了した最終ルートです。検疫ルートは、相互の再帰が解消したか確認するために定期的に評価されます。再帰が引き続き存在する場合は、ルートは検疫対象のままとなります。再帰が解消した場合は、ルートは検疫対象から外れます。

次の手順を使用して、ルートを隔離します。

1. RIB は問題がある特定のパスがインストールされている場合に検出します。
2. RIB は、そのパスを取り込んだプロトコルに通知を送信します。
3. プロトコルは問題のルートに関する隔離通知を受信すると、そのルートを「隔離中」とマークします。これが BGP ルートである場合、BGP はそのネイバーにルートへの到達可能性をアドバタイズしません。
4. RIB は、すべての検疫対象パスに対して、安全に取り込む（検疫対象から「使用 OK」状態に移行）ことができるようになったかどうかを定期的にテストします。パスが安全に使用できるようになったことを示す通知がプロトコルに送信されます。

ルートとラベルの整合性チェッカ

ルート整合性チェッカおよびラベル整合性チェッカ（RCC/LCC）はコマンドライン ツールです。これは、コントロール プレーンとデータ プレーン ルート間および IOS XR ソフトウェアのラベル プログラミングの整合性を検証するために使用できます。

運用中ネットワークのルータは、転送情報がコントロール プレーン情報と一致しない状態になった可能性があります。この原因は、ルート プロセッサ（RP）とラインカード（LC）間でのファブリック障害または転送障害、または転送情報ベース（FIB）に関する問題である可能性があります。RCC/LCC を使用すると、結果として生じたコントロール プレーンとデータ プレーン間の不整合を識別して詳細情報を出力できます。この情報は、転送問題とトラフィック損失の原因をさらに調査して診断するために使用できます。

RCC/LCC は、2 つのモードで実行できます。RCC/LCC は、適切なコマンド モードをオンデマンドとして使用して 1 回かぎりのスキャンでトリガーする（オンデマンド スキャン）か、通常のルータ動作中にバックグラウンドで定義した間隔で実行するように設定（バックグラウンド スキャン）できます。RCC は、ルーティング情報ベース（RIB）を転送情報ベース（FIB）と比較します。一方、LCC は、ラベル スイッチング データベース（LSD）を FIB と比較します。不整合が検出されると、RCC/LCC 出力では、特定のルートまたはラベルを識別し、検出された不整合のタイプを識別して、さらなるトラブルシューティングに役立つ追加のデータも提供します。

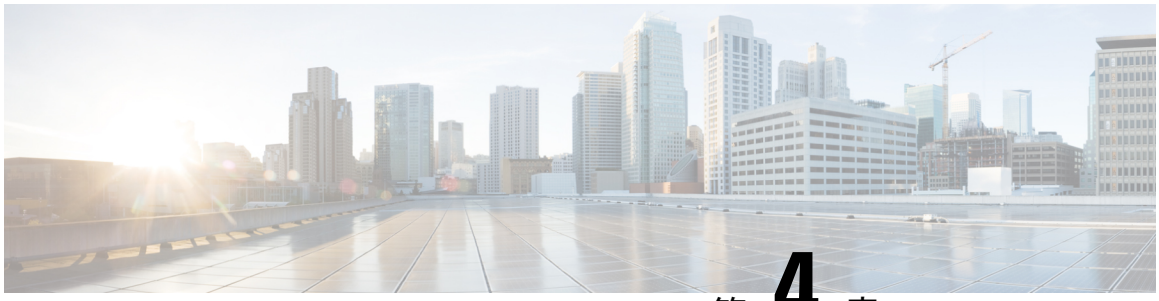
RCC はルート プロセッサで動作します。FIB は、ライン カード上のエラーについてチェックし、最初の 20 のエラー レポートを RCC に送信します。RCC はすべてのノードからエラー レポートを受信し、それらを要約し（完全一致についてチェックし）、2 つのキュー（ソフトまたはハード）に追加します。各キューのエラー レポート数の制限は 1000 で、キューに優先度はありません。RCC/LCC は、異なるノードからの同じエラー（完全一致）を 1 つのエラーとして記録します。RCC/LCC は、エラーのプレフィックス/ラベル、バージョン番号、タイプなどに基づいてエラーを比較します。

オンデマンド スキャン

オンデマンド スキャンでは、ユーザは、特定のテーブルの特定のプレフィックスまたはテーブル内のすべてのプレフィックス関するコマンドライン インターフェイス全体のスキャンを要求します。スキャンはただちに実行され、結果がすぐに発行されます。LCC は LSD でオンデマンド スキャンを実行するのに対し、RCC は VRF 単位で実行します。

バックグラウンド スキャン

バックグラウンド スキャンでは、ユーザはバックグラウンドで実行されるスキャンを設定します。設定は、定期的なスキャンの間隔で構成されます。このスキャンは、単一または複数のテーブルに設定できます。LCC は LSD でバックグラウンド スキャンを実行するのに対し、RCC はデフォルトの VRF または他の VRF に対し実行します。



第 4 章

ルーティング ポリシーの実装

ピアから受け入れるか、ピアにアドバタイズされる、または1個のルーティングプロトコルから別のプロトコルへ再配布されるときに、ルートを検査し、フィルタリングして、属性を変更するように、ルーティング ポリシーがルータに指示します。

このモジュールでは、ルーティング プロトコルが設定済みのルーティング ポリシーに基づいて、ルートのアドバタイズ、集約、廃棄、配布、エクスポート、保留、インポート、再配布、変更を決定する方法について説明します。

ルーティング ポリシー言語（RPL）では、すべてのルーティング ポリシーのニーズを表現できる、単一の直接的な言語です。RPLは、大規模なルーティング設定をサポートするように設計されました。以前のルーティング ポリシー コンフィギュレーション方式に固有の冗長性が大幅に削減されました。RPL では、ルーティング ポリシーの設定が簡素化され、これらの設定の保存および処理に必要なシステム リソースが削減され、トラブルシューティングが容易になりました。

（注）

- 現在は、デフォルトの VRF のみがサポートされています。L3VPN、VPNv4、VPNv6 およびVPNルーティング/転送（VRF）のアドレス ファミリおよびマルチキャストは、今後のリリースでサポートされる予定です。

- ルーティング ポリシー実装の制約事項 （127 ページ）
- ルート ポリシーの定義 （128 ページ）
- BGP ネイバーへのルーティング ポリシーのアタッチ （130 ページ）
- テキスト エディタを使用したルーティング ポリシーの変更 （131 ページ）
- ルーティング ポリシーの参照 （134 ページ）

ルーティング ポリシー実装の制約事項

次の制約事項は、ルーティング ポリシー言語の実装と連動している場合に適用されます。

- ボーダー ゲートウェイ プロトコル（BGP）、Integrated Intermediate System-to-Intermediate（IS-IS）または Open Shortest Path First（OSPF）がネットワーク内で設定されている必要があります。
- 最大 1000 のステートメントのポリシー定義がサポートされています。ポリシー内のステートメントの総数は、階層型ポリシー構造を使用して 4000 ステートメントに拡張できます。ただし、この制限は、**apply** ステートメントの使用に限定されます。
- 接続点で直接または間接的に付加されたポリシーを変更する必要がある場合、単一の **commit** 操作は次の場合に実行できません。
  - 接続点に直接または間接的に付加された別のポリシーによって参照されているセットまたはポリシーを削除する。
  - 削除するものと同じセットまたはポリシーへの参照を削除するためにポリシーを変更する。

  commit は、次の 2 つの手順で実行する必要があります。
  1. ポリシーまたはセットへの参照を削除するためにポリシーを変更してから、**commit** を実行します。
  2. ポリシーまたはセットを削除してから、**commit** を実行します。
- 内部および外部の BGP マルチパスが設定されている Carrier Supporting Carrier（CSC）ネットワークでは、vrf 単位のラベル モードはサポートされていません。

ルート ポリシーの定義

ここでは、ルート ポリシーを定義する方法について説明します。

（注）
- Command-Line Interface（CLI）を使用して既存のルーティング ポリシーを変更する場合、このタスクを実行してポリシーを再定義する必要があります。
- RPL のスケール設定の変更には、時間がかかる場合があります。
- BGP は、大規模な RPL 設定の変更が原因でクラッシュしたり、または連続した RPL の変更時にクラッシュする可能性があります。BGP のクラッシュを回避するには、以降の変更をコミットする前に、BGP In/Out キュー内にメッセージがなくなるまで待機します。

手順

ステップ 1 **configure**

ステップ 2 **route-policy** *name* [*parameter1* , *parameter2* , . . . , *parameterN*]

例：

```
RP/0/RP0/cpu 0: router(config)# route-policy sample1
```

ルートポリシー コンフィギュレーション モードを開始します。

- ルート ポリシーの開始後、ルート ポリシーを定義する一連のコマンドを入力できます。

ステップ 3 **end-policy**

例：

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

ルート ポリシーの定義を終了して、ルート ポリシー コンフィギュレーション モードを終了します。

ステップ 4 **commit**

ルーティング ポリシー定義：例

次の例では、route-policy name コマンドを使用して、sample1 という BGP ルート ポリシーが定義されます。ポリシーはネットワーク層到達可能性情報（NLRI）をプレフィックス セット test 内の要素と比較します。真と評価されると、ポリシーは *then* 句の中の操作を実行します。偽と評価されると、ポリシーは *else* 句の中の操作を実行します。つまり、MED 値を 200 とし、ルートにコミュニティ 2:100 を追加します。例の最終段階では、コンフィギュレーションをルータにコミットし、コンフィギュレーション モードを終了してルート ポリシー sample1 の内容を表示します。

```
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
```

BGP ネイバーへのルーティング ポリシーのアタッチ

このタスクでは、ルーティング ポリシーの BGP ネイバーへのアタッチ方法を説明します。

始める前に

ルーティングポリシーは、接続点に適用される前に、設定を済ませよく定義しておく必要があります。ポリシーが事前に設定されていない場合、ポリシーが定義されていないことを示すエラー メッセージが生成されます。

手順

ステップ 1 **configure**

ステップ 2 **router bgp** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config)# router bgp 125
```

BGP ルーティング プロセスを設定し、ルータ コンフィギュレーション モードを開始します。

- *as-number* 引数は、ルータが存在する自律システムを識別します。有効値は、0 ～ 65535 です。内部ネットワークで使用できるプライベート自律システム番号の範囲は、64512 ～ 65535 です。

ステップ 3 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 10.0.0.20
```

ネイバー IP アドレスを指定します。

ステップ 4 **address-family** { **ipv4 unicast** | | **ipv6 unicast** | } **address-family** { **ipv4** | **ipv6** } **unicast**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# address-family ipv4 unicast
```

アドレス ファミリを指定します。

ステップ 5 **route-policy** *policy-name* { **in** | **out** }

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# route-policy example1 in
```

ルート ポリシーをアタッチします。事前に作成と定義を済ませておく必要があります。

ステップ 6 **commit**

テキストエディタを使用したルーティングポリシーの変更

このタスクでは、テキスト エディタを使用して既存のルーティング ポリシーを変更する方法を説明します。

手順

ステップ 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*]

例：

```
RP/0/RP0/cpu 0: router# edit route-policy sample1
```

変更するルート ポリシー、プレフィックス セット、AS パス セット、コミュニティ セット、または拡張コミュニティ セットの名前を指定します。

- ルート ポリシー、プレフィックス セット、AS パス セット、コミュニティ セット、または拡張コミュニティセットのコピーが一時ファイルとして作成され、エディタが起動します。
- Nano での編集後に、エディタ バッファを保存し、Ctrl+X キーストロークを使用してエディタを終了します。
- Emacs での編集後に、Ctrl+X および Ctrl+S のキーストロークを使用して編集バッファを保存します。エディタを保存して終了するには、Ctrl+X および Ctrl+C のキーストロークを使用します。
- Vim での編集後に、現在のファイルに書き込んで終了するには、:wq、:x、または ZZ のキーストロークを使用します。終了して確認するには、:q キーストロークを使用します。終了して変更を廃棄するには、:q! キーストロークを使用します。

ステップ 2 **show rpl route-policy** [*name* [**detail**] | **states** | **brief**]

例：

```
RP/0/RP0/cpu 0: router# show rpl route-policy sample2
```

（任意）特定の名前付きルート ポリシーのコンフィギュレーションを表示します。

- ポリシーが使用するすべてのポリシーとセットを表示するには **detail** キーワードを使用します。

- 未使用、非アクティブ、アクティブ状態のものをすべて表示するには **states** キーワードを使用します。
- すべての拡張コミュニティ セットの名前を設定なしでリストするには、**brief** キーワードを使用します。

ステップ 3 **show rpl prefix-set** [*name* | **states** | **brief**]

例：

```
RP/0/RP0/cpu 0: router# show rpl prefix-set prefixset1
```

（任意）名前付きプレフィックス セットの内容を表示します。

- 名前付き AS パス セット、コミュニティ セット、拡張コミュニティ セットの内容を表示するには **prefix-set** キーワードをそれぞれ **as-path-set** 、 **community-set** 、または **extcommunity-set** にそれぞれ置き換えます。

シンプル インバウンド ポリシー：例

次のポリシーは、ネットワーク層到達可能性情報（NLRI）が/24よりも長いプレフィックスを指定するルート、およびNLRIがRFC 1918によって予約されているアドレス空間の宛先を指定するルートを廃棄します。残りのルートすべてに対しては、MEDとローカル プリファレンスを設定し、ルートのリストにコミュニティを追加します。

コミュニティ リストに101:202～106:202の範囲の値を含み、値202を含む16ビットタグ部分を持つルートの場合、ポリシーは、自律システム番号2を先頭に2回追加し、ルートのリストにコミュニティ2:666を追加します。これらのルートに対して、MEDが666または225のいずれかである場合、ポリシーはルートの送信元を不完全に設定し、それ以外の場合はIGPに設定します。

コミュニティ リストが範囲101:202～106:202内の値を含まない場合、ポリシーはルート内のリストにコミュニティ2:999を追加します。

```
prefix-set too-specific
 0.0.0.0/0 ge 25 le 32
end-set

prefix-set rfc1918
 10.0.0.0/8 le 32,
 172.16.0.0/12 le 32,
 192.168.0.0/16 le 32
end-set

route-policy inbound-tx
 if destination in too-specific or destination in rfc1918 then
  drop
 endif
 set med 1000
 set local-preference 90
 set community (2:1001) additive
```

```
 if community matches-any ([101..106]:202) then
  prepend as-path 2.30 2
  set community (2:666) additive
 if med is 666 or med is 225 then
  set origin incomplete
 else
  set origin igp
 endif
else
  set community (2:999) additive
 endif
 end-policy

 router bgp 2
  neighbor 10.0.1.2 address-family ipv4 unicast route-policy inbound-tx in
```

次のポリシーの例に、2 つの異なるピア向けに 2 つのインバウンド ポリシー in-100 と in-101 の作成方法を示します。それらのピアに特定のポリシーを作成するとき、ポリシーは複数のピアに共通する可能性があるポリシーの共通ブロックを再利用します。いくつかの基本的な構築ブロックとして、policies common-inbound、filter-bogons、set-lpref-prepend が作成されます。

filter-bogons 構築ブロックは、RFC 1918 アドレス空間からのルートといった不要なルートをフィルタするシンプルなポリシーです。ポリシー set-lpref-prepend は、渡されるパラメータ化された値に応じて、ローカル プリファレンスを設定し、その先頭に AS パスを追加できるユーティリティ ポリシーです。common-inbound ポリシーは、これらの filter-bogons 構築ブロックを使用して、インバウンド ポリシーの共通ブロックを構築します。common-inbound ポリシーは、in-100 と in-101 作成の構築ブロックとして、set-lpref-prepend 構築ブロックとともに使用されます。

```
  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
```

ルーティング ポリシーの参照

RPL を実装するには、次の概念を理解する必要があります。

ルーティング ポリシー言語

ここでは、次の内容について説明します。

ルーティング ポリシー言語の概要

RPLは、大規模なルーティング設定をサポートするように開発されました。RPLには、従来のルート マップ、アクセス リスト、プレフィックス リスト向けの設定の機能とは異なる、重要な機能がいくつか備えられています。これらの機能の1つめは、モジュラ形式でポリシーを構築する機能です。ポリシーの共通ブロックは、個別に定義および維持できます。次に、ポリシーのこれらの共通ブロックをポリシーの他のブロックから適用して完全なポリシーを構築できます。この機能により、維持する必要のある設定情報の量が減ります。また、ポリシーのこれらの共通ブロックはパラメータ化できます。パラメータ化により、同じ構造を共有するが、設定または一致された特定の値が異なるポリシーをポリシーの独立したブロックとして維持できます。たとえば、ローカルプリファレンス値以外はすべて同一である3つのポリシーは、ポリシーのパラメータとして異なるローカルプリファレンス値を持つ1つの共通のパラメータ化ポリシーとして表せます。

このポリシー言語では、セットという概念が導入されました。セットとは、ルート属性の一致および設定演算で使用できる類似したデータのコンテナです。セットタイプには、prefix-sets、community-sets、as-path-sets、およびextcommunity-setsの4つがあります。これらのセットはそれぞれ、IPv4またはIPv6プレフィックス、コミュニティ値、ASパス正規表現、および拡張コミュニティ値のグループ化を保持します。セットは、データの単なるコンテナです。セットの

大半はインライン変数も保持します。インラインセットでは、名前付きセットを参照せずに、値の短い列挙をポリシーで直接使用できます。プレフィックスリスト、コミュニティリスト、および AS パス リストは、リストに項目が 1 つか 2 つしかない場合でも維持が必要です。RPL のインラインセットを使用すると、名前付きセットを参照することなく、値の小さいセットをポリシー本体に直接配置できます。

許可や拒否などの決定は、ポリシー定義自体で明示的に制御されます。RPL は、セット データを使用する可能性のある一致演算子を、従来のブール論理演算子 AND、OR、および NOT と組み合わせて複雑な条件式にします。すべての一致演算は true または false の結果を返します。その後、これらの条件式の実行および関連するアクションは、*if then*、*elseif*、および *else* の単純な構造を使用して制御できます。これにより、ポリシーを通じて評価パスをすべて指定できます。

ルーティング ポリシー言語の構造

ここでは、RPL の基本構造について説明します。

名前

ポリシー言語には、セットとポリシーの 2 種類の持続的で名前を付けられるオブジェクトがあります。これらのオブジェクトの定義は、開始および終了のコマンドラインとして括弧で囲まれます。たとえば、test という名前のポリシーを定義する設定構文は、次のようになります。

```
route-policy test
[ . . . policy statements . . . ]
end-policy
```

ポリシー オブジェクトの正規名は、任意の連続する英数字（大文字および小文字）、数字の 0 ～9、および句読文字のピリオド、ハイフン、およびアンダースコアで指定できます。名前は、文字または数字ではじまる必要があります。

セット

このコンテキストでは、セットという用語を、順序付けのない固有のエレメントの集合を意味する数学的な概念で使用されます。ポリシー言語は、セットをマッチング用の値のグループに対するコンテナとして提供します。セットは、条件式で使用されます。セットの要素はカンマで区切ります。ヌル（空）のセットは許可されます。

次の例で、

```
prefix-set backup-routes
  # currently no backup routes are defined
end-set
```

次の条件は、

```
if destination in backup-routes then
```

すべてのルートに対して FALSE として評価されます。これは、プレフィックス セットに一致する一致条件がないためです。

たとえば、2 つまたは 3 つのコミュニティ値のように少ない数の要素に対して比較を実行する場合があるとします。これらの比較を実現するため、ユーザはこれらの値を直接列挙できます。これらの列挙はインライン セットと呼ばれます。インライン セットは、機能的には名前付きセットと同じですが、単純なテストをインラインにできるようにします。つまり、1 つや 2 つだけの要素を比較する際には、別の名前付きセットの維持を比較では必要としません。構文については、次の項で説明するセット型を参照してください。通常、インラインセットの構文は、括弧で囲まれたカンマ区切りのリストです。ここで element-entry は、プレフィックスやコミュニティ値などの使用タイプに適切な項目のエントリです。

次に、インライン コミュニティ セットを使用する例を示します。

```
route-policy sample-inline
if community matches-any ([10..15]:100) then
set local-preference 100
endif
end-policy
```

次に、test-communities という名前付きセットを使用する同等の例を示します。

```
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
```

これらの両方のポリシーは機能的に同等ですが、インライン形式では、6 つの値を格納するだけのためにコミュニティ セットの設定を必要としません。設定コンテキストに適切な形式を選択できます。次の各項では、名前付きセットバージョンとインライン形式の両方の例が必要に応じて示されています。

as-path-set

AS パス セットは、AS パス属性と一致させるための演算で構成されます。一致演算は、正規表現一致だけです。

名前付きセット形式

名前付きセット形式は、**ios-regex** キーワードを使用して正規表現のタイプを示します。また、正規表現を単一引用符で囲む必要があります。

次の例では、名前付き AS パス セットの定義を示します。

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

この AS パス セットは 2 つの要素から構成されます。一致演算で使用する場合は、この AS パスセットは、ASパスが自律システム（AS）番号42または127のいずれかで終わる任意のルートと一致します。

名前付き AS パス セットを削除するには、**no as-path-set aset1** コマンドライン インターフェイス（CLI）コマンドを使用します。

（注） 正規表現一致により、CPU の負荷が高くなります。ポリシー パフォーマンスを大幅に向上するには、次のいずれかを実行します。正規表現パターンをまとめて正規表現の合計呼び出し数を減らす、または「as-path neighbor-is」、「as-path originates-from」、「as-path passes-through」などの同等のネイティブ as-path 一致演算を使用します。

インライン セット形式

インライン セットは、次のようにカンマ区切りの式のリストを括弧で囲んだ形式です。

```
(ios-regex '_42$', ios-regex '_127$')
```

このセットは、前の名前付きセットと同じ AS パスと一致させますが、ポリシーが使用する名前付きセットとは別に名前付きセットを作成する余計な作業を必要としません。

community-set

コミュニティ セットは、BGP コミュニティ属性との一致のためにコミュニティ値を保持しています。コミュニティは、32 ビット量です。整数のコミュニティ値は半分に分けて、コロンで区切った、0 ～ 65535 の範囲内の 2 つの符号なし 10 進整数で表す必要があります。単一の 32 ビットコミュニティ値は指定できません。次に、名前付きセット形式を示します。

名前付きセット形式

```
community-set cset1
12:34,
12:56,
12:78,
internet
```

```
end-set
```

インライン セット形式

```
(12:34, 12:56, 12:78)
($as:34, $as:$tag1, 12:78, internet)
```

コミュニティセットのインライン形式では、パラメータ化もサポートされます。コミュニティの 16 ビット部分のそれぞれをパラメータ化できます。

RPL では、標準の well-known コミュニティ値のシンボル名が提供されます。internet は 0:0、no-export は 65535:65281、no-advertise は 65535:65282、local-as は 65535:65283 です。

RPL では、コミュニティの指定でワイルドカードを使用するためのファシリティも用意されています。ワイルドカードを指定するには、コミュニティ指定の 16 ビット部分の 1 つの代わりに、アスタリスク（*）を挿入します。ワイルドカードはコミュニティのその部分の任意の値が一致することを示します。つまり、次のポリシーが一致するすべてのコミュニティの中で、自律システムが属するコミュニティは 123 です。

```
community-set cset3
   123:*
end-set
```

すべてのコミュニティセットに、少なくとも 1 つのコミュニティ値が含まれている必要があります。空のコミュニティ セットは無効なため拒否されます。

extcommunity-set

拡張コミュニティセットは、通常のコミュニティ値の代わりに拡張コミュニティ値が含まれている点を除き、コミュニティ セットと似ています。拡張コミュニティ セットは、名前付き形式およびインライン形式もサポートします。拡張コミュニティ セットには、cost、soo、rt の 3 つのタイプがあります。

コミュニティセットと同様に、インライン形式では、パラメータ化ポリシー内のパラメータ化がサポートされます。拡張コミュニティ値のいずれかの部分をパラメータ化できます。

ワイルドカード（*）および正規表現は、拡張コミュニティ セット要素で使用できます。

すべての拡張コミュニティセットに、少なくとも 1 つの拡張コミュニティ値が含まれている必要があります。空の拡張コミュニティ セットは無効なため拒否されます。

次に、構文例を示します。

extcommunity-set rt の名前付き形式

rt セットは BGP ルート ターゲット（RT）拡張コミュニティ タイプのコミュニティを格納するために使用される extcommunity セットです。

```
extcommunity-set rt a_rt_set
```

```
    1.2.3.4:666
    1234:666,
    1.2.3.4:777,
    4567:777
  end-set

  Inline Set Form for Extcommunity-set RT

  (1.2.3.4:666, 1234:666, 1.2.3.4:777, 4567:777)
  ($ipadrr:666, 1234:$tag, 1.2.3.4:777, $tag2:777)
```

次のオプションが拡張コミュニティ セット RT でサポートされています。

```
RP/0/RP0/cpu 0: router(config)#extcommunity-set rt rt_set
RP/0/RP0/cpu 0: 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
```

オプション	説明
#-remark	「#」ではじまる注記
*	ワイルドカード（任意のコミュニティまたはその一部）
<1-4294967295>	32 ビットの 10 進数
<1-65535>	16 ビットの 10 進数
A.B.C.D/M:N	拡張コミュニティ：IPv4 プレフィックス形式
A.B.C.D:N	拡張コミュニティ：IPv4 形式
ASN:N	拡張コミュニティ：AS プレーン形式
X.Y:N	拡張コミュニティ：ASDOT 形式
abort	RPL 定義を廃棄して、top level config に戻る
dfa-regex	DFA スタイルの正規表現
end-set	セット定義の終了
exit	このサブモードを終了
ios-regex	従来の IOS スタイルの正規表現

オプション	説明
show	一部の RPL 設定を表示

extcommunity-set soo の名前付き形式

soo セットは、BGP Site-of-Origin（SoO）拡張コミュニティ タイプ コミュニティを格納するために使用される extcommunity セットです。

```
  extcommunity-set soo a_soo_set
1.1.1:100,
     100:200
  end-set
```

次のオプションが拡張コミュニティ セット soo でサポートされています。

```
RP/0/RP0/cpu 0: router(config)#extcommunity-set soo soo_set
RP/0/RP0/cpu 0: 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
```

オプション	説明
#-remark	「#」ではじまる注記
*	ワイルドカード（任意のコミュニティまたはその一部）
<1-4294967295>	32 ビットの 10 進数
<1-65535>	16 ビットの 10 進数
A.B.C.D/M:N	拡張コミュニティ：IPv4 プレフィックス形式
A.B.C.D:N	拡張コミュニティ：IPv4 形式
ASN:N	拡張コミュニティ：AS プレーン形式
X.Y:N	拡張コミュニティ：ASDOT 形式
abort	RPL 定義を廃棄して、top level config に戻る
dfa-regex	DFA スタイルの正規表現

オプション	説明
end-set	セット定義の終了
exit	このサブモードを終了
ios-regex	従来の IOS スタイルの正規表現
show	一部の RPL 設定を表示

prefix-set

prefix-set は、それぞれ 4 つの部分（アドレス、マスク長、最小一致長、最大一致長）がある IPv4 または IPv6 プレフィックス一致指定を保持しています。アドレスは必須ですが、他の 3 つの部分は任意です。アドレスは、標準のドット付き IPv4 またはコロンで区切られた 16 進数の IPv6 アドレスです。マスク長（存在する場合）は、0 ～ 32（IPv6 の場合は 0 ～ 128）の範囲内の負以外の 10 進整数で、その前のアドレスはスラッシュで区切ります。アドレスと任意のマスク長の後には、任意の最小一致長が続き、これはキーワード **ge**（以上（**g**reater than or **e**qual to）のニーモニック）で表され、その後に 0 ～ 32（IPv6 の場合は 0 ～ 128）の範囲内の負以外の 10 進整数が続きます。最後には、任意の最大一致長が続き、これはキーワード **le**（以下（**l** ess than or **e**qual to）のニーモニック）で表され、その後に 0 ～ 32（IPv6 の場合は 0 ～ 128）の範囲内の負以外の別の 10 進整数が続きます。一致させるプレフィックスの正確な長さを指定するための構文ショートカットは、**eq** キーワード（等しい（**eq**ual to）のニーモニック）です。

プレフィックス一致指定にマスク長がない場合は、デフォルトのマスク長は、IPv4 では 32、IPv6 では 128 です。デフォルトの最小マッチング長はマスク長です。最小一致長が指定されている場合、デフォルトの最大一致長は、IPv4 では 32、IPv6 では 128 です。最小と最大のいずれも指定しない場合は、デフォルトの最大長はマスク長になります。

prefix-set 自体は、プレフィックス一致指定のカンマ区切りのリストです。次に例を示します。

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

prefix-set legal-ipv6-prefix-examples
  2001:0:0:1::/64,
  2001:0:0:2::/64 ge 96,
  2001:0:0:2::/64 ge 96 le 100,
  2001:0:0:2::/64 eq 100
end-set
```

prefix-set の最初の要素は、唯一の有効値 10.0.1.1/32 またはホスト アドレス 10.0.1.1 と一致します。2 番めの要素は、唯一の有効値 10.0.2.0/24 と一致します。3 番めの要素は、10.0.3.0/28 ～ 10.0.3.255/32 の範囲のプレフィックス値と一致します。4 番めの要素は、10.0.4.0/24 ～

10.0.4.240/28 の範囲の値と一致します。5 番めの要素は、10.0.5.0/26 ～ 10.0.5.252/30 の範囲内のプレフィックスと一致します。6 番めの要素は、10.0.6.0/28 ～ 10.0.6.240/28 の範囲内にある長さ 28 の任意のプレフィックスと一致します。7 番目の要素は、10.0.[0..255].2/32（10.0.0.2/32 ～ 10.0.255.2）の範囲内にある長さ 32 の任意のプレフィックスと一致します。8 番目の要素は、10.[0..255].8.0/26（10.0.8.0/26 ～ 10.255.8.0/26）の範囲内にある長さ 26 の任意のプレフィックスと一致します。

次の prefix-set はすべて、無効なプレフィックス一致指定からなります。

```
prefix-set ILLEGAL-PREFIX-EXAMPLES
   10.1.1.1 ge 16,
   10.1.2.1 le 16,
   10.1.3.0/24 le 23,
   10.1.4.0/24 ge 33,
   10.1.5.0/25 ge 29 le 28
end-set
```

最小長と最大長のいずれも、マスク長がないと無効です。IPv4 の場合、最小長は、IPv4 プレフィックスの最大長である 32 未満でなければなりません。IPv6 の場合、最小長は、IPv6 プレフィックスの最大長である 128 未満でなければなりません。最大長は、最小長以上でなければなりません。

RPL プレフィックス セットでの *ACL* サポート

アクセス コントロール リスト（ACL）タイプのプレフィックス セット エントリは、IPv4 または IPv6 のプレフィックス一致指定を保持し、それぞれにアドレスとワイルドカード マスクがあります。アドレスおよびワイルドカードマスクは、標準のドット付き 10 進数表記の IPv4 アドレスまたはコロンで区切られた 16 進数の IPv6 アドレスです。照合するビット セットはワイルドカードの形式（反転マスクとも呼ばれる）で提供され、バイナリ 0 は必須一致を、バイナリ 1 は一致しない条件をそれぞれ表します。プレフィックス セットを使用して、任意のルートで一致する必要がある連続したビット セットと非連続のビット セットを指定できます。

rd-set

rd-set は、ルート識別子（RD）要素を含むセットを作成するために使用します。RD セットは、固有のボーダー ゲートウェイ プロトコル（BGP）VPN IPv4 アドレスをグローバルに作成するために、IPv4 アドレスが前に付いた 64 ビット値です。

RD 値は、次のコマンドを使用して定義できます。

- *a.b.c.d:m:**：IPv4 形式の BGP VPN RD とワイルドカード文字。たとえば、10.0.0.2:255.255.0.0:* です。
- *a.b.c.d/m:n*：IPv4 形式の BGP VPN RD とマスク。たとえば、10.0.0.2:255.255.0.0:666 です。
- *a.b.c.d:***：IPv4 形式の BGP VPN RD とワイルドカード文字。たとえば、10.0.0.2:255.255.0.0 です。
- *a.b.c.d:n*：IPv4 形式の BGP VPN RD。たとえば、10.0.0.2:666 です。

- *asn:**：ASN 形式の BGP VPN RD とワイルドカード文字。たとえば、10002:255.255.0.0 です。
- *asn:n*：ASN 形式の BGP VPN RD。たとえば、10002:666 です。

次に、rd-set の例を示します。

```
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
```

ルーティング ポリシー言語コンポーネント

ルーティング ポリシー言語の 4 種類の主要コンポーネント、設定フロント エンド、ポリシー リポジトリ、実行エンジン、およびポリシー クライアントそのものは、ポリシーの定義、変更、および使用に関係します。

設定フロントエンド（CLI）は、ポリシーを定義し、変更する機能です。この設定は、通常の保存方法を使用してルータに保存され、通常の設定の**show** コマンドを使用して表示できます。

ポリシー インフラストラクチャの 2 番目のコンポーネントである、ポリシー リポジトリには複数の役割があります。最初に、ユーザ入力設定を実行エンジンが認識可能な形式にコンパイルします。2 番目に、ポリシー検証の多くを実行し、定義されたポリシーが実際に適切に実行できることを確認します。3 番目に、いずれの接続点がいずれのポリシーを使用しているかを追跡して、ポリシーが変更された場合に適切なクライアントが適切な新しいポリシーで正常にアップデートされるようにします。

3 番目のコンポーネントは実行エンジンです。このコンポーネントは、クライアントの要求に応じて実際にポリシーを実行する部分です。このプロセスは、ポリシークライアントのいずれかからルートを受信して、特定のルートデータに対して実際のポリシーを実行するまでと見なせます。

4 つめのコンポーネントはポリシークライアント（ルーティングプロトコル）です。このコンポーネントは、適切なタイミングで実行エンジンをコールし、特定のポリシーを特定のルートに適用し、次にいくつかのアクションを実行します。これらのアクションには、ルートをドロップする必要があるとポリシーに示されている場合にルートを削除する、最適なルートの候補としてプロトコル決定ツリーにルートを渡す、またはポリシーに変更されたルートを必要に応じてネイバーまたはピアにアドバタイズすることが含まれます。

ルーティング ポリシー言語使用方法

ここでは、基本的なルーティング ポリシー言語の使用方法の例について説明します。

パス ポリシー パス ポリシー

次に、ルートを変更せずにポリシーがすべての渡されたルートを受け入れる例を示します。

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

すべてをドロップするポリシー

次に、渡されたすべてのルートをポリシーが明示的に拒否する例を示します。このタイプのポリシーは、特定のピアから送信されるすべてを無視するために使用されます。

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

パスの特定の AS 番号を使用するルートを無視する

次に、3 個の部分からなるポリシー定義の例を示します。まず、as-path-set コマンドは、AS パスとマッチングするための 3 つの正規表現を定義します。2 番目に、route-policy コマンドは、ルートに AS パス セットを適用します。ルートの AS パス属性が as-path-set コマンドで定義された正規表現と一致する場合、プロトコルはこのルートを拒否します。3 番めに、ルート ポリシーは、BGP ネイバー 10.0.1.2 に付加されます。BGP は、ネイバー 10.0.1.2 から（インポート）受信したルートの ignore_path_as という名前のポリシーを参照します。

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

route-policy ignore_path_as
if as-path in ignore_path then
drop
else
pass
endif
end-policy

router bgp 2
neighbor 10.0.1.2 address-family ipv4 unicast policy ignore_path_as in
```

MED に基づくセット コミュニティ

次に、ポリシーがルートのMEDをテストし、MEDの値に基づいてルートのコミュニティ属性を変更する例を示します。MED 値が 127 の場合、ポリシーはルートにコミュニティ 123:456 を追加します。MED 値が 63 の場合、ポリシーはルートのコミュニティ属性に値 123:789 を追加します。それ以外の場合、ポリシーはコミュニティ 123:123 をルートから削除します。いかなる場合でも、ルートを受け入れるようにポリシーからプロトコルに指示します。

```
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
```

コミュニティに基づくローカル プリファレンスの設定

次に、quickstart-communities という名前のコミュニティ セットがコミュニティ値を定義する例を示します。quickstart-localpref という名前のルート ポリシーは、quickstart-communities コミュニティ セットに指定されたコミュニティの存在に対してルートをテストします。ルートにいずれかのコミュニティ値が存在する場合、ルート ポリシーはルートのローカル プリファレンス属性を 31 に設定します。いかなる場合でも、ルートを受け入れるようにポリシーからプロトコルに指示します。

```
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
```

持続的な注記

次に、ポリシーのセットおよびステートメントのエントリの意味を明確にするために、ポリシーにコメントを配置する例を示します。注記は持続的なため、ポリシーに付加されたままです。たとえば、show running-config コマンドの出力には注記が表示されます。ポリシーに注記を追加すると、ポリシーの理解、また後日の変更が容易になるとともに、予期しない動作が発生した場合にトラブルシューティングが容易になります。

```
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
```

ポリシー定義

ポリシー定義によって、ポリシー ステートメントの名前付きシーケンスが作成されます。ポリシー定義は、CLI の **route-policy** キーワードと、その後に続く名前、ポリシー ステートメントのシーケンス、および **end-policy** キーワードで構成されます。たとえば、次のポリシーは検出されたルートをすべてドロップします。

```
route-policy drop-everything
drop
end-policy
```

名前は、ポリシーをプロトコルにバインドするためのハンドルとして機能します。ポリシー定義を削除するには、no route-policy name コマンドを発行します。

ポリシーの共通ブロックを再利用できるように、ポリシーは他のポリシーを参照していることもあります。他のポリシーの参照は、**apply** ステートメントを使用して、次の例のように実行されます。

```
route-policy check-as-1234
if as-path passes-through ‘1234.5’ then
apply drop-everything
else
pass
endif
end-policy
```

apply ステートメントは、検討中のルートが受信前に自律システム 1234.5 を介してパスルーされた場合に、ポリシー drop-everything を実行する必要があることを示しています。AS パスに自律システム 1234.5 があるルートが受信された場合、ルートはドロップされます。それ以外の場合、ルートは変更なしで受け入れられます。このポリシーは、階層型ポリシーの例です。つまり、**apply** ステートメントのセマンティックは、適用されるポリシーを切り取って適用するポリシーに貼り付けた場合と同様です。

```
route-policy check-as-1234-prime
 if as-path passes-through '1234.5' then
    drop
 else
    pass
 endif
end-policy
```

必要に応じて階層レベルはいくつでも使用できます。しかし、レベルを多くすると、維持や理解が困難になることがあります。

パラメータ化

apply ステートメントを使用したポリシーの再利用サポートに加えて、属性の一部のパラメータ化ができるようにポリシーを定義できます。次に、param-example という名前のパラメータ化ポリシーを定義する例を示します。この場合、ポリシーは $mytag という 1 つのパラメータを取ります。パラメータは、常にドル記号ではじまり、任意の英数字で構成されます。パラメータは、パラメータを取る属性に置き換えられます。

次の例では、16 ビット コミュニティ タグがパラメータとして使用されています。

```
route-policy param-example ($mytag)
set community (1234:$mytag) additive
end-policy
```

その後、このパラメータ化ポリシーは、次の例に示すように、別のパラメータ化で再利用できます。この方法で、共通の構造を共有するが、一部の個別のステートメントで別の値を使用するポリシーをモジュール化できます。パラメータ化できる属性の詳細については、個別の属性についての項を参照してください。

```
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
```

パラメータ化ポリシー param-example は、apply ステートメントのパラメータとして提供されている値で拡張されたポリシー定義を提供します。ポリシー階層が常に維持されるため、param-example の定義が変更されると、origin_10 および origin_20 の動作が一致するように変更されることに注意してください。

origin-10 ポリシーの効果として、このポリシーをパススルーし、自律システム 10 から発信されたルートを示す AS パスを持つすべてのルートにコミュニティ 1234:10 を追加します。origin-20 ポリシーは同様ですが、自律システム 20 から発信されたルートに対してコミュニティ 1234:20 を追加します。

接続点でのパラメータ化

apply ステートメントを使用したパラメータ化のサポートに加えて、ポリシーは接続点での属性のパラメータ化ができるようにも定義できます。すべての接続点で、パラメータ化がサポートされます。

次の例では、パラメータ化ポリシー「param-example」を定義します。この例では、ポリシーは、「$mymed」と「$prefixset」の 2 つのパラメータを取ります。パラメータは、常にドル記号ではじまり、任意の英数字で構成されます。パラメータは、パラメータを取る属性に置き換えられます。この例では、MED 値およびプレフィックス セット名をパラメータとして渡しています。

```
route-policy param-example ($mymed, $prefixset)
  if destination in $prefixset then
  set med $mymed
  endif
end-policy
```

その後、このパラメータ化ポリシーは、次の例に示すように、別のパラメータ化で再利用できます。この方法で、共通の構造を共有するが、一部の個別のステートメントで別の値を使用するポリシーをモジュール化できます。パラメータ化できる属性の詳細については、各プロトコルの個別の属性を参照してください。

```
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)
```

パラメータ化ポリシー param-example は、neighbor route-policy in and out ステートメントのパラメータとして提供されている値で拡張されたポリシー定義を提供します。

グローバル パラメータ化

RPL では、ポリシー定義内で使用できるシステム全体のグローバル パラメータの定義がサポートされます。グローバル パラメータは、次のように設定できます。

```
Policy-global
  glbpathtype ‘ebgp’
  glbtag ‘100’
end-global
```

グローバル パラメータ値は、パラメータ化ポリシーのローカル パラメータと類似したポリシー定義内で直接使用できます。次の例では、*globalparam* 引数は、グローバル パラメータ gbpathtype と glbtag を使用し、非パラメータ化ポリシーに対して定義されます。

```
route-policy globalparam
  if path-type is $glbpathtype then
    set tag $glbtag
  endif
end-policy
```

パラメータ化ポリシーのパラメータ名に、グローバル パラメータ名との「衝突」がある場合は、ポリシー定義に対してローカルなパラメータが優先され、グローバルパラメータが効果的に隠されます。さらに、特定のグローバルパラメータが任意のポリシーによって参照されている場合は、削除されないように、検証メカニズムが実施されます。

ポリシー適用のセマンティック

ここでは、ルーティングポリシーの評価および適用方法について説明します。説明する概念は次のとおりです。

ブール演算子優先

ブール式は、演算子優先順位に従って、左から右に評価されます。最も優先される演算子は NOT であり、その後に AND、次に OR と続きます。次に、式の例を示します。

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

評価の順を表示するために、すべてが括弧で囲まれる場合は、次のようになります。

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

内部の NOT は、宛先のテストのみに適用されます。AND は、NOT 式の結果を Multi Exit Discriminator（MED）テストと組み合わせます。さらに、OR は、その結果をコミュニティ テストと組み合わせます。演算の順は配列し直される場合があります。

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

その場合、すべてが括弧で囲まれた式は、次のようになります。

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

同じ属性の複数の変更

ポリシーが属性の値を複数回置換する場合、すべてのアクションが実行されるため、最後の割り当てが優先されます。BGP の MED 属性は、1 つの固有値であるため、それに設定された最後の値が優先されます。つまり、次のポリシーでは、ルートの MED 値は 12 になります。

```
set med 9
set med 10
set med 11
set med 12
```

この例は単純ですが、機能の場合は複雑です。属性の値を効率的に変更するポリシーを記述することも可能です。次に例を示します。

```
set med 8
if community matches-any cs1 then
set local-preference 122
if community matches-any cs2 then
set med 12
endif
endif
```

結果として、ルートの MED は 8 になります。ただし、ルートのコミュニティ リストが cs1 と cs2 の両方と一致する場合は、結果として、ルートの MED は 12 になります。

変更している属性に1つの値だけが含められる場合は、最後のステートメントが優先されるため、この例を理解するのは簡単です。一方、複数の値を含められる属性もいくつかあり、このような属性における複数のアクションの結果は、置換ではなく累積します。最初の例として、コミュニティおよび拡張コミュニティ評価で **additive** キーワードを使用する場合があります。形式のポリシーを検討します。

```
route-policy community-add
set community (10:23)
set community (10:24) additive
set community (10:25) additive
end-policy
```

このポリシーは、10:23、10:24、および 10:25 の 3 つのコミュニティ値すべてを含めるためにルートにコミュニティ ストリングを設定します。

2番目の例として、ASパスの先頭に追加する場合があります。形式のポリシーを検討します。

```
route-policy prepend-example
prepend as-path 2.5 3
prepend as-path 666.5 2
end-policy
```

このポリシーは、AS パスの先頭に 666.5 666.5 2.5 2.5 2.5 を追加します。この先頭に追加する動作は、実行するすべてのアクションの結果であり、単一のスカラー値ではなく、値の配列を含む属性となる AS パスに対して実行されます。

属性を変更するとき

ポリシーは、すべてのテストが完了するまでルート属性値を変更しません。つまり、比較演算子はルートの初期データで常に実行されます。ルート属性の中間での変更は、ポリシーの評価にカスケード効果を与えません。次に、例を示します。

```
ifmed eq 12 then
set med 42
if med eq 42 then
```

```
drop
endif
endif
```

このポリシーは、drop ステートメントを一切実行しません。これは、2 番目のテスト（med eq 42）がルートの元の変更されていない MED 値を見るためです。2 番目のテストに到達するには、MED が 12 である必要があるため、2 番目のテストは常に false を返します。

デフォルトのドロップ処理

すべてのルート ポリシーには、評価中のルートをドロップするデフォルト アクションがあります。ただし、ルートがポリシーアクションによって変更された場合と明示的に渡された場合は除きます。適用（ネスト）されるポリシーは、適用されるポイントに適用されるポリシーを貼り付けることによってこの処理を実装します。

ネットワーク 10 のすべてのルートを許可し、そのローカルプリファレンスを 200 に設定して、他のルートすべてをドロップするポリシーについて検討します。ポリシーは次のように記述できます。

```
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
```

明示的な **pass** ステートメントが含まれていなく、ルート属性も変更しないため、ポリシー one はすべてのルートをドロップするように見える可能性があります。しかし、適用されるポリシーは実際は一部のルートに属性を設定し、この処理はポリシー one に渡されます。結果として、ポリシー one はネットワーク 10 の宛先を含むルートを渡して、その他すべてをドロップします。

制御フロー

ポリシーステートメントは、設定に表示される順に従って順番に処理されます。他のポリシーブロックを階層的に参照するポリシーは、参照されるポリシーブロックが直接インラインに置換されたように処理されます。たとえば、次のポリシーが定義されているとします。

```
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
```

ポリシー four は次と同様の方法で書き換えられます。

```
route-policy four-equivalent
set weight 100
set med 200
set community (2:666) additive
pass
end-policy
```

（注） **pass** ステートメントは必要ないため削除して、別の方法で同様のポリシーを表せます。

ポリシー検証

ポリシーが定義されて使用される際には、いくつかの異なるタイプの検証が発生します。

範囲チェック

ポリシーが定義されるとき、値の範囲チェックなどのいくつかの簡単な検証が行われます。たとえば、設定される MED が MED 属性の適切な範囲内にあることを検証するためにチェックされます。しかし、この範囲チェックはパラメータ指定を対象にできません。これは、パラメータ指定が値をまだ定義していない可能性があるためです。パラメータ指定は、ポリシーが接続点に付加されたときに検証されます。ポリシーリポジトリでも、ポリシーの再帰定義がなく、パラメータの数値が正しいことが検証されます。付加時には、すべてのポリシーが正しく形成されている必要があります。参照されるすべてのセットおよびポリシーが定義されていて、有効値を持っている必要があります。同様に、パラメータ値もすべて適切な範囲内にある必要があります。

不完全なポリシーとセットの参照

指定のポリシーが接続点に付加されていないかぎり、ポリシーは存在しないセットおよびポリシーを参照できます。これにより、ワークフローが自由になります。まだ定義されていないセットまたはポリシーブロックを参照する設定を構築して、後からこれらの未定義のポリシーおよびセットに入力できます。これにより、ポリシー定義でのより高い柔軟性を実現します。参照するポリシーの各部分は、ポリシーを定義しているときに設定に存在する必要はありません。つまり、ユーザはポリシーバーが存在しない場合でも、**apply** ステートメントを使用してポリシーバーを参照するポリシー例を定義できます。同様に、ユーザは、存在しないセットを参照するポリシー ステートメントを入力できます。

ただし、参照されているすべてのポリシーとセットの存在は、ポリシーが付加されたときに適用されます。neighbor 1.2.3.4 address-family ipv4 unicast policy sample in コマンドを使用してインバウンド BGP ポリシーで未定義ポリシー バーを参照するポリシー サンプルをアタッチしようとすると、ポリシー バーが存在しないために設定が拒否されます。

同様に、接続点で現在使用中のルートポリシーまたはセットは削除できません。これは、削除の結果、未定義参照が発生するためです。現在使用中のルートポリシーまたはセットを削除しようとすると、ユーザに対してエラー メッセージが表示されます。

ポリシーバーが存在しているものの、ステートメント、アクション、ディスポジションが存在しないヌル ポリシーと呼ばれる条件が存在します。つまり、ポリシー バーは次の場合に存在します。

```
route-policy bar
end-policy
```

これは、有効なポリシーブロックです。これは、ルートを一切変更せず、pass ステートメントも含まれていないポリシーブロックであるため、すべてのルートのドロップを実質的に強制します。したがって、ポリシー ブロックのドロップのデフォルト アクションが実行されます。

集約

aggregation 接続点は、その集約のサブコンポーネントの条件付き存在に基づいてアドバタイズされる集約ルートを生成します。この接続点で付加されるポリシーでは、有効な BGP 属性であればどれでも集約ルートに設定できます。たとえば、生成される集約にポリシーによってコミュニティ値や MED を設定できます。名前付きポリシーによって評価されるルートのいずれかがポリシーをパスする場合、指定された集約が生成されます。集約の詳細は、**suppress-route** キーワードを使用してフィルタ処理されます。ルート内の属性を設定するアクションは、すべて集約上の属性に影響します。

ポリシー言語では、コンフィギュレーションはポリシーをパスするルートによって制御されます。summary-only フラグが設定されていない場合、抑制マップを使用して集約の特定のコンポーネントに対する選択的なフィルタリングや抑制を行います。つまり、集約およびさらに特定のコンポーネントが送出される場合、さらに特定のコンポーネントの一部を抑制マップを使用してフィルタできます。ポリシー言語では、ルートの選択と suppress フラグの設定によってこれを制御します。attribute-map を使用して、ユーザは集約されたルート上で特定の属性を設定できます。ポリシー言語では、集約されたルート上の属性設定は通常のアクション操作によって制御されます。

次の例では、範囲 10.0.0.0/8 ge 8 le 25（10.2.0.0/24 を除く）にルーティングされたコンポーネントがある場合、集約アドレス 10.0.0.0/8 が生成されます。summary-only が設定されていないため、集約のコンポーネントすべてがアドバタイズされます。しかし、特定のコンポーネント 10.1.0.0 は抑制されます。

```
route-policy sample
  if destination in (10.0.0.0/8 ge 8 le 25) then
    set community (10:33)
  endif
  if destination in (10.2.0.0/24) then
```

```
    drop
  endif
  if destination in (10.1.0.0/24) then
    suppress-route
  endif
end-policy

router bgp 2
address-family ipv4
  aggregate-address 10.0.0.0/8 route-policy sample
  .
  .
  .
```

集約の属性上の集約ポリシーの効果は累積します。集約ポリシーがより具体的なルートにマッチするたびに、ポリシー内の設定動作が集約を修正します。次の例の集約は、集約を構成する、より具体的なルートの数に応じて、異なる MED 値を持ちます。

```
route-policy bumping-aggregation
  set med +5
end-policy
```

3 種類のより具体的なルートにマッチした場合、集約の MED はデフォルトに 15 を加算したものになります。17 種類のより具体的なルートの場合、MED はデフォルトに 85 を加算したものになります。

集約ポリシーがプレフィックス パスに適用される順序は決定論的ですが、未指定です。つまり、あるルートのセットはいつでも同様の順序で出現しますが、順序を予測する方法はありません。

集約ポリシーのドロップが集約の生成を妨げることはありませんが、現在のより具体的なルートが集約に寄与できなくなります。別のより具体的なルートがルートにパスを渡す場合、集約が生成されます。集約の生成に必要なのは、より具体的なパス 1 つだけです。

ポリシー ステートメント

ポリシー ステートメントには、注記、処理（drop および pass）、アクション（set）、および if（比較演算子）の 4 つのタイプがあります。

注記

注記は、ポリシー設定に添付されているテキストですが、それ以外の点ではポリシー言語パーサーによって無視されるテキストです。注記は、備考はポリシーの一部を記述するのに役立ちます。注記の構文は、各行の先頭にポンド記号（#）があるテキストです。

```
# This is a simple one-line remark.

# This
# is a remark
# comprising multiple
# lines.
```

通常、注記はセットの完全なステートメントまたは要素の間で使用されます。ステートメントの途中、またはインライン セット定義内での注記はサポートされていません。

CLI における従来の！コメントとは異なり、RPL の注記は、リブートしても持続し、また設定がディスクまたは TFTP サーバに保存されてからルータにロードされても持続します。

処理

ポリシーがルートを変更した場合、デフォルトではポリシーは、そのルートを受け入れます。RPL には、その反対を強制する **drop** ステートメントがあります。ポリシーがルートに一致してドロップを実行する場合、ポリシーはルートを受け入れません。ポリシーがルートを変更しない場合、デフォルトでそのルートはドロップされます。ルートのドロップを防止するには、**pass** ステートメントを使用します。

drop ステートメントは、ルートを破棄するアクションを実行するように指示します。ルートがドロップされると、ポリシーはこれ以上実行されません。たとえば、ポリシーの最初の2つのステートメントを実行した後で、**drop** ステートメントが検出されると、ポリシーは停止してルートが破棄されます。

（注） すべてのポリシーにおいて、実行の最後にはデフォルトの **drop** アクションがあります。

pass ステートメントを使用すると、ルートが変更されなかった場合でもポリシーは実行を継続できます。ポリシーの実行が終了すると、ポリシーで変更されたか、ポリシーで pass 処理を受信したルートはすべてポリシーを渡し、実行は完了します。ルート ポリシー B_rp がルート ポリシー A_rp 内に適用されたとき、プレフィックスがポリシー B_rp によってドロップされない場合は、実行はポリシー A_rp からポリシー B_rp へと続けられてから、ポリシー A_rp に戻ります。

```
route-policy A_rp
  set community (10:10)
  apply B_rp
end-policy
!

route-policy B_rp
  if destination in (121.23.0.0/16 le 32, 155.12.0.0/16 le 32) then
  set community (121:155) additive
  endif
end-policy
!
```

ポリシーがルート属性を**変更する**か、ポリシーが明示的な **pass** ステートメントによってルートを渡さないかぎり、デフォルトでルートはポリシー処理の最後に**ドロップされます**。たとえば、ルート ポリシー B がルート ポリシー A 内に適用されたとき、プレフィックスがポリシー B によってドロップされない場合は、実行はポリシー A からポリシー B へと続けられてから、ポリシー A に戻ります。

```
route-policy A
   if as-path neighbor-is '123' then
   apply B
   policy statement N
end-policy
```

一方で、次のポリシーは評価するすべてのルートを渡します。

```
route-policy PASS-ALL
pass
end-policy
```

```
route-policy SET-LPREF
set local-preference 200
end-policy
```

黙示的にドロップされることに加えて、ルートは**明示的な drop** ステートメントによってドロップされることもあります。**drop** ステートメントによってルートがすぐにドロップされるため、ポリシー処理はこれ以上実行されません。また、**drop** ステートメントは、それ以前に処理された **pass** ステートメントまたは属性変更をすべて上書きすることにも注意してください。たとえば、次のポリシーはすべてのルートをドロップします。最初の **pass** ステートメントは実行されますが、すぐに **drop** ステートメントによって上書きされます。2 番目の **pass** ステートメントが実行されることは決してありません。

```
route-policy DROP-EXAMPLE
pass
drop
pass
end-policy
```

1 つのポリシーが別のポリシーを適用する場合、適用されるポリシーは適用するポリシーの正しい位置にコピーされたようになり、次に、同じ drop-and-pass セマンティックが施行されます。たとえば、次のポリシー ONE および TWO は、ポリシー ONE-PRIME と同等です。

```
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
```

明示的な drop ステートメントの効果が即時であるため、10.0.0.0/16 le 32 内のルートは、これ以上のポリシー処理なしでドロップされます。次に、その他のルートが自律システム 123 によってアドバタイズされているかどうかを確認するために検討されます。アドバタイズされている場合、それらのルートは渡されます。それ以外の場合は、すべてのポリシー処理の最後に黙示的にドロップされます。

done ステートメントは、ポリシーの実行を停止してルートを受け入れるアクションを実行するように指示します。**done** ステートメントを検出すると、ルートが渡され、ポリシー ステートメントはこれ以上実行されません。**done** ステートメントの前にルートに対して行った変更はすべて、有効なままです。

アクション

アクションとは、ルートを変更する基本操作のシーケンスです。すべてではありませんが、大部分のアクションは **set** キーワードによって区別されます。ルート ポリシーでは、アクションはグループ化できます。次に、3 つのアクションから構成される 1 つのルート ポリシーの例を示します。

```
route-policy actions
set med 217
set community (12:34) additive
delete community in (12:56)
end-policy
```

If

最も単純な形式では、**if** ステートメントは、条件式を使用して、指定のルートで行う必要があるアクションまたは処理を決定します。次に例を示します。

```
if as-path in as-path-set-1 then
drop
endif
```

この例では、AS パスがセット as-path-set-1 にあるすべてのルートがドロップされることを示しています。**then** 句の内容には、ポリシー ステートメントの任意のシーケンスが指定できます。

次の例では、2 つのアクション ステートメントが含まれています。

```
if origin is igp then
set med 42
prepend as-path 73.5 5
endif
```

CLI では、endif コマンドに代わる、exit コマンドがサポートされています。

if ステートメントでは、if 条件が false の場合に実行される **else** 句も使用できます。

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

ポリシー言語では、テストのシーケンスをつなぎ合わせるために、**elseif** キーワードを使用した構文も用意されています。

```
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
```

次の例に示すように、**if** ステートメント内のステートメント自体が **if** ステートメントになることがあります。

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

このポリシーの例では、コミュニティ値 12:34 または 56:78 が関連付けられたすべてのルートで、ローカルプリファレンス属性の値が 100 に設定されます。ただし、これらのルートのいずれかで MED 値が 150 になっている場合は、コミュニティ値 12:34 または 56:78 のいずれかおよび MED 150 が指定されたこれらのルートはドロップされます。

ブール条件

if ステートメントについて説明した前の項では、すべての例で true または false を評価する単純なブール条件を使用しています。RPL には、ブール演算子を使用して、単純条件から複合条件を構築する方法もあります。

ブール演算子には、否定（**not**）、論理積（**and**）、および論理和（**or**）の 3 つがあります。ポリシー言語では、否定が最も優先され、その後に論理積、次に論理和と続きます。優先を上書きする、または可読性を高める目的で複合条件をグループ化するために括弧を使用できます。

次に、単純条件の例を示します。

```
med eq 42
```

ルートの MED の値が 42 の場合は true、それ以外の場合は false です。

単純条件は、**not** 演算子を使用しても否定できます。

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

括弧で囲まれているブール条件は、それ自体がブール条件です。

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

複合条件は次の 2 つの形式のいずれかになります。複合条件は、単純式の後に **and** 演算子が続くか、それ自体の後に単純条件が続きます。

```
med eq 42 and next-hop in (10.0.2.2)
```

複合条件は、単純式の後に **or** 演算子、またその後に別の単純条件が続くこともできます。

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

複合条件全体を括弧で囲むこともできます。

```
(med eq 42 and next-hop in (10.0.2.2))
```

括弧は、サブ条件のグループの可読性を高めるために使用される場合、またはサブ条件を 1 つのユニットとして評価するように強制する場合があります。

次の例では、最優先の **not** 演算子は、宛先のテストのみに適用されます。**and** 演算子は、**not** 式の結果をコミュニティ テストと組み合わせます。また、**or** 演算子は、その結果を MED テストと組み合わせます。

```
med eq 10 or not destination in (10.1.3.0/24) and community matches-any
([12..34]:[56..78])
```

優先を表すために一連の括弧を使用すると、次のようになります。

```
med eq 10 or ((not destination in (10.1.3.0/24)) and community matches-any
([12..34]:[56..78])
```

次に、複雑な式の別の例を示します。

```
(origin is igp or origin is incomplete or not med eq 42) and next-hop in (10.0.2.2)
```

左の論理積は、括弧で囲まれた複合条件です。複合条件内部の最初の単純条件は、送信元属性の値をテストします。値が内部ゲートウェイ プロトコル（IGP）の場合、複合条件内部は true です。それ以外の場合、評価は再び送信元属性の値のテストに進み、これが不完全な場合は複合条件内部はtrueです。それ以外の場合、評価は次の成分条件（単純条件の否定）のチェックに進みます。

apply

ポリシー定義およびポリシーのパラメータ化の項で説明したように、**apply** コマンドは別のポリシー（パラメータ化または未パラメータ化のいずれか）を別のポリシー内から実行します。これにより、ポリシーの共通ブロックの再利用が可能になります。ポリシーの共通ブロックのパラメータ化機能と併用した場合、**apply** コマンドは、設定の繰り返しを軽減するための強力なツールになります。

接続点

ポリシーはルートに適用されるまで有用になりません。ルーティングプロトコルがルートに適用されるポリシーを知る必要があります。たとえば、BGPではポリシーが使用されるいくつかの状況がありますが、最も一般的なのはインポートおよびエクスポートポリシーの定義です。ポリシー接続点は、特定のプロトコル エンティティ（この場合 BGP ネイバー）と特定の名前付きポリシーとのアソシエーションが形成されるポイントです。検証手順がこのポイントで発生することに留意してください。あるポリシーがアタッチされるたびに、そのポリシーとそれが適用される可能性があるすべてのポリシーについて、そのポリシーを接続点で有効に使用できるか確かめられます。たとえば、ユーザが IS-IS レベル属性を設定するポリシーを定義し、このポリシーをインバウンド BGP ポリシーとしてアタッチしようとする場合、BGP ルートは IS-IS 属性を持たないため、この操作は拒否されます。同様に、使用中のポリシーが変更される場合、そのポリシーの現在の使用すべてについて、変更内容が現在の使用と互換性があるかどうかが検証されます。

各プロトコルは、それぞれルートを構成する属性（コマンド）のセットの定義を持っています。たとえば、BGPルートにOSPFで未定義のコミュニティ属性が存在する場合があります。IS-IS内のルートには、BGPには未知のレベル属性があります。RIBで内部的に保持されるルートは、タグ属性を持つ場合があります。

あるプロトコルにポリシーがアタッチされると、プロトコルはそのポリシーがプロトコルにとって既知のルート属性を使用して動作するものであるか確認します。プロトコルが未知の属性を使用している場合、プロトコルはアタッチを拒否します。たとえば、OSPFはBGPコミュニティの値をテストするポリシーのアタッチを拒否します。

各プロトコルは最低2つの異なるルートタイプにアクセスできるため、状況はさらに複雑になります。ネイティブ プロトコル ルートに加え、BGP または IS-IS を例にすると、一部のプロトコル ポリシー接続点は RIB ルート上で動作し、これは共通の中心的表現です。BGP を例にすると、プロトコルは RIB から BGP に再配布されたルートへポリシーを適用する接続点を提

供します。2 つの異なる種類のルートを処理する接続点では、マッチングには RIB 属性の動作、設定には BGP 属性の動作というように、動作の混在が許可されます。

（注） プロトコル コンフィギュレーションは、サポートされていない動作を実行するポリシーの付加を拒否します。

続く項では、プロトコル接続点について、各接続点で有効な属性（コマンド）および動作に関する情報を含めて説明します。

BGP ポリシー接続点

この項では、それぞれの BGP ポリシー接続点について説明し、BGP 属性と演算子の概要を示します。

Additional-Path

additional-path 接続点を使用して、さまざまな属性のマッチング演算に基づいた、いっそう詳細な制御が行えます。この接続点は、BGP スピーカーがプレフィックスに対して複数のパスを送信できるように追加パスを選択するために、ルートポリシーを使用するかどうかを決定するために使用されます。

追加パスにより、エッジルータでの BGP プレフィックス独立コンバージェンス（PIC）が可能になります。

次に、追加パス選択のイネーブル化に使用するルート ポリシー「add-path-policy」を設定する例を示します。

```
router bgp 100
 address-family ipv4 unicast
 additional-paths selection route-policy add-path-policy
```

Default Originate

default originate 接続点により、デフォルト ルート（0.0.0.0/0）を条件に応じて生成し、他のルートの存在に基づいてピアにアドバタイズできます。このコンフィギュレーションは、Routing Information Base（RIB）に対して関連付けられたポリシーの評価によって実行されます。ポリシーをパスするルートがある場合、デフォルト ルートが生成され、関連ピアに送られます。

次のポリシーでは、RIB 内に 10.0.0.0/8 ge 8 le 32 にマッチするルートが存在する場合に、デフォルト ルートを生成して BGP ネイバー 10.0.0.1 に送ります。

```
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

neighbor export 接続点は、特定のピアまたはピアのグループに送信する BGP ルートを選択します。このルートは、関連するポリシー全体に考えられる BGP ルートのセットを実行することによって選択されます。ポリシーをパスするすべてのルートが、ピアまたはピア グループにアップデートとして送信されます。送信されるルートは、適用されているポリシーによってその BGP 属性が変更されている場合があります。

次のポリシーは、すべての BGP ルートをネイバー 10.0.0.5 に送ります。2:100 から 2:200 までの範囲内の任意のコミュニティ タグが付けられたルートは、MED の値が 100、コミュニティの値が 2:666 で送信されます。その他のルートは、MED の値が 200、コミュニティの値が 2:200 で送信されます。

```
route-policy sample-export
  if community matches-any (2:[100-200]) then
    set med 100
    set community (2:666)
  else
    set med 200
    set community (2:200)
  endif
end-policy

router bgp 2
  neighbor 10.0.0.5
    remote-as 3
  address-family ipv4 unicast
    route-policy sample-export out
    .
    .
    .
```

Neighbor Import

neighbor import 接続点は、指定ピアからのルートの受け入れを制御します。ピアから受け取るすべてのルートは、アタッチされたポリシーについて実行されます。付加されたポリシーを渡すルートは、最適なパス ルート選択の候補として BGP Routing Information Base（BRIB）に渡されます。

BGP インポート ポリシーが変更されると、そのピアから受け取ったルートすべてを新しいポリシーに対して再実行することが必要になります。変更されたポリシーは、それまで全体に許可されていたルートを廃棄し、それまで廃棄されていたルートを全体に許可します。または、ルートが修正された方法を変更します。BGP の新しいコンフィギュレーション オプション（**bgp auto-policy-soft-reset**）によって、ソフト再設定が実行されるか BGP ルート リフレッシュ機能がネゴシエートされた場合に、この変更を自動的に発生させられます。

次の例は、ネイバー 10.0.0.1 からルートを受け取る方法を示しています。コミュニティ 3:100 で受け取ったルートは、ローカル プリファレンスが 100 に設定され、コミュニティ タグは

2:666 に設定されます。このピアからのその他のルートはすべて、ローカル プリファレンスが 200 に、コミュニティ タグが 2:200 に設定されます。

```
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

network 接続点は RIB から BGP へのルートの挿入を制御します。このポイントでアタッチされるルート ポリシーは、挿入されるルートのどの有効な BGP 属性でも設定できます。

次の例では、/24 よりも具体的なすべてのルートに対して well-known コミュニティ no-export を設定するネットワーク接続点にアタッチされたルート ポリシーを表示します。

```
route-policy NetworkControl
  if destination in (0.0.0.0/0 ge 25) then
    set community (no-export) additive
  endif
end-policy

router bgp 2
  address-family ipv4 unicast
    network 172.16.0.5/27 route-policy NetworkControl
```

Redistribute

redistribute 接続点は、他のソースからのルートを BGP がアドバタイズできるようにします。このポイントでアタッチされるポリシーは、再配布されるルートのどの有効な BGP 属性でも設定できます。同様に、ユーザは選択演算子によって再配布されるルート送信元およびそれら送信元からのルートを制御できます。

次の例に、OSPF インスタンス 12 から BGP へのすべてのルートを再配布する方法を示します。デフォルトのルートを持つ OSPF はドロップされます。タグ 10 を持つルートは、ローカル プリファレンスが 300 に、コミュニティ値が 2:666 に設定され、no-advertise がアタッチされます。他のルートはすべて、ローカル プリファレンスが 200 に、コミュニティ値が 2:100 にセットされます。

```
route-policy sample_redistribute
  if destination in (0.0.0.0/0) then
```

```
    drop
  endif
  if tag eq 10 then
     set local-preference 300
     set community (2:666, no-advertise)
  else
    set local-preference 200
    set community (2:100)
  endif
end-policy

router bgp 2
  address-family ipv4 unicast
    redistribute ospf 12 route-policy sample_redistribute
    .
    .
```

Show BGP

show bgp 接続点を使用して、指定のポリシーを渡す、選択した BGP ルートを表示できます。アタッチされたポリシーによってドロップされていないルートが、show bgp コマンドによる出力に似た形式で表示されます。

次の例では、show bgp route-policy コマンドを使用して、MED の値が 5 の BGP ルートを表示します。

```
route-policy sample-display
  if med eq 5 then
    pass
  endif
end-policy
!
show bgp route-policy sample-display
```

show bgp policy route-policy コマンドも存在しており、RIB がアウトバウンド BGP ポリシーであるかのようにして、名前付きポリシーを通過した RIB 内のすべてのルートを実行します。このコマンドは、次の例に示すように、各ルートが変更の前と後にどのようになったかを表示します。

```
show rpl route-policy test2

route-policy test2
  if (destination in  (10.0.0.0/8 ge 8 le 32)) then
    set med 333
  endif
end-policy
!

show bgp

BGP router identifier 10.0.0.1, local AS number 2
BGP main routing table version 11
BGP scan interval 60 secs
Status codes:s suppressed, d damped, h history, * valid, > best
             i - internal, S stale
Origin codes:i - IGP, e - EGP, ? - incomplete
```

```
   Network            Next Hop            Metric LocPrf Weight Path
*> 10.0.0.0           10.0.1.2                10             0 3 ?
*> 10.0.0.0/9         10.0.1.2                10             0 3 ?
*> 10.0.0.0/10        10.0.1.2                10             0 3 ?
*> 10.0.0.0/11        10.0.1.2                10             0 3 ?
*> 10.1.0.0/16        10.0.1.2                10             0 3 ?
*> 10.3.30.0/24       10.0.1.2                10             0 3 ?
*> 10.3.30.128/25     10.0.1.2                10             0 3 ?
*> 10.128.0.0/9       10.0.1.2                10             0 3 ?
*> 10.255.0.0/24      10.0.101.2            1000    555      0 100 e
*> 10.255.64.0/24     10.0.101.2            1000    555      0 100 e
....
```

show bgp policy route-policy test2

```
10.0.0.0/8 is advertised to 10.0.101.2

  Path info:
    neighbor:10.0.1.2        neighbor router id:10.0.1.2
    valid  external  best
  Attributes after inbound policy was applied:
    next hop:10.0.1.2
    MET ORG AS
    origin:incomplete  neighbor as:3  metric:10
    aspath:3
  Attributes after outbound policy was applied:
    next hop:10.0.1.2
    MET ORG AS
    origin:incomplete  neighbor as:3  metric:333
    aspath:2 3
...
```

Table Policy

table policy 接続点を使用して、グローバル ルーティング テーブルにインストールされるときのルートのトラフィック索引の値を設定できます。この接続点は、BGPポリシーアカウンティング機能をサポートします。BGP ポリシー アカウンティングは、BGP ルートに設定されたトラフィック索引を使用してさまざまなカウンタをトラックします。このように、ルータ演算子は、マッチング演算を使用して、BGPルート属性の複数の異なるセットを選択し、トラッキング対象のルートのクラスそれぞれに異なるトラフィック インデックスを設定できます。

次に、自律システム 10.33 を始点とする IPv4 ユニキャスト ルートでトラフィック索引を 10 に設定する例を示します。同様に、自律システム 11.60 を始点とする IPv4 ユニキャスト ルートのトラフィック索引は、FIB にインストールされるときに 11 に設定されます。これらトラフィック索引は、対象インターフェイス上でBGP ポリシー アカウンティング カウンタを有効にすることで、ルートインラインカード上で転送されるトラフィックをカウントするために使用されます。

```
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

import 接続点を使用して、グローバル VPN IPv4 テーブルから特定の VPN ルーティングおよび転送（VRF）インスタンスへのルートのインポートを制御できます。

レイヤ 3 VPN ネットワークの場合、Provider Edge（PE）ルータは他の PE ルータから Multiprotocol Internal Border Gateway Protocol（MP-iBGP）を通じて VPN IPv4 ルートを学習し、その VRF のインポート ルート ターゲットにマッチするルート ターゲットを含まないルート通知を自動的にフィルタリングして除外します。

この自動ルート フィルタリングは RPL コンフィギュレーションなしで発生します。ただし、VRF でのルートのインポートをより詳細に制御するために、VRF インポート ポリシーを設定できます。

次の例に、ルート ターゲット拡張コミュニティに基づいたマッチングおよびそれに続くネクスト ホップ設定の実行方法を示します。ルートのルート ターゲット値が 10:91 が設定されている場合、ネクスト ホップは 172.16.0.1 に設定されます。ルートのルート ターゲット値が 11:92 が設定されている場合、ネクスト ホップは 172.16.0.2 に設定されます。ルートの Site of Origin（SoO）値が 10:111111 または 10:111222 の場合、ルートはドロップされます。一致しないすべてのルートはドロップされます。

```
route-policy bgpvrf_import
  if extcommunity rt matches-any (10:91) then
    set next-hop 172.16.0.1
  elseif extcommunity rt matches-every (11:92) then
    set next-hop 172.16.0.2
  elseif extcommunity soo matches-any (10:111111, 10:111222) then
    pass
  endif
end-policy

vrf vrf_import
  address-family ipv4 unicast
    import route-policy bgpvrf_import
    .
    .
    .
```

Export

export 接続点は、特定の VRF からグローバル VPN IPv4 テーブルへのルートのエクスポート全体を制御します。

レイヤ 3 VPN ネットワークでは、VRF IPv4 ルートが VPN IPv4 ルートに変換され MP-iBGP 経由で他の PE ルータへアドバタイズされる場合（またはある VRF から PE ルータ内の他の VRF へ流れる場合）、エクスポート ルート ターゲットは VPN IPv4 ルートに追加されます。

エクスポート ルート ターゲットのセットは RPL コンフィギュレーションなしで VRF に設定されます。ただし、条件に応じてルート ターゲットを設定するために、VRF エクスポート ポリシーを設定できます。

エクスポート ルート ポリシー用にサポートされているマッチングと設定の操作例を次に示します。あるルートが 172.16.1.0/24 にマッチした場合、ルート ターゲット拡張コミュニティは 10:101 に、重みは 211 に設定されます。ルートが 172.16.1.0/24 にマッチしないもののその始点が egp である場合、ローカル プリファレンスが 212 に、ルート ターゲット拡張コミュニティは 10:101 に設定されます。指定された条件にマッチしないルートの場合、ルート ターゲット拡張コミュニティ 10:111222 がルートに追加されます。さらに、前記の条件のいずれかにマッチするルートにも RT 10:111222 が追加されます。

```
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

BGP 内の retain route target 接続点を使用して、ルート ターゲット拡張コミュニティだけに基づいたマッチング条件を指定できます。この接続点は、Route Reflector（RR）または Autonomous System Boundary Router（ASBR）で役立ちます。

通常、RR はその PE ルータとのピアのためにすべての IPv4 VPN ルートを保持しておく必要があります。これら PE は、異なるルート ターゲット IPv4 VPN ルートでタグ付けされたルータを要求する場合があり、結果として RR が拡張不能になります。ルート ターゲット拡張コミュニティの定義済みセット、およびサービスする VPN の特定のセットを持つルートを RR が保持するよう設定することで、拡張性が得られます。

この接続点を使用する別の理由は、ASBR のためです。ASBR では VRF を設定する必要はありませんが、このコンフィギュレーションによって IPv4 VPN プレフィックス情報を保持する必要があります。

次の例に、ルート ポリシー リテイナーを設定し、retain route target 接続点に適用する方法を示します。ルート ターゲット拡張コミュニティ 10:615、10:6150、15.15.15.15.15:15 を含むルートが受け入れられます。一致しないすべてのルートはドロップされます。

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

```
  15.15.15.15.15:15
  end-set

route-policy retainer
  if extcommunity rt matches-any rtset1 then
    pass
  endif
end-policy

router bgp 2
  address-family vpnv4 unicast
    retain route-target route-policy retainer
    .
    .
    .
```

Allocate-Label

allocate-label 接続点を使用して、さまざまな属性のマッチング操作に基づいた、いっそう詳細な制御が行えます。この接続点は、IPv4 ラベル付きユニキャスト アドレス ファミリ用にアップデートをネイバーに送信するときに、inter-AS オプション C 内のラベル割り当てが必要かどうか判断するために使用されます。サポートされている属性設定アクションは、パスとドロップです。

Label-Mode

ラベルモード接続点では、プレフィックス値、コミュニティなどの任意の一致基準に基づいてラベル モードを選択する機能が提供されます。この接続点は、通常、ラベル モードのタイプを、展開環境設定に基づいて per-ce または per-vrf または per-prefix に設定するために使用されます。サポートされている属性設定アクションは、パスとドロップです。

Neighbor-ORF

neighbor-orf 接続点によって、プレフィックス ベースのマッチングだけを使用した着信 BGP ルート アップデートのフィルタリングが行えます。インバウンド フィルタとしての使用に加え、フィルタリングを実行できるよう、Outbound Route Filter（ORF）としてプレフィックスと処理内容（ドロップまたはパス）が上流ネイバーに送られます。

次の例に、ルート ポリシー orf-preset を設定し、ネイバー ORF 接続点に適用する方法を示します。orf-preset で指定されたプレフィックス（172.16.1.0/24、172.16.5.0/24、172.16.11.0/24）にマッチした場合、ルートのプレフィックスはドロップされます。ネイバーが宛先に送信する前にフィルタ更新を行うことができるように、BGP も、このインバウンド フィルタリングのほかに、許可または拒否とともにこれらのプレフィックス エントリをアップストリーム ネイバーに送信します。

```
prefix-set orf-preset
  172.16.1.0/24,
  172.16.5.0/24,
  172.16.11.0/24
end-set

route-policy policy-orf
  if orf prefix in orf-preset then
    drop
  endif
```

```
  if orf prefix in (172.16.3.0/24, 172.16.7.0/24, 172.16.13.0/24) then
    pass
  endif

router bgp 2
  neighbor 1.1.1.1
    remote-as 3
    address-family ipv4 unicast
      orf route-policy policy-orf
    .
    .
    .
```

Next-hop

next-hop 接続点を使用して、より詳細なプロトコル ベースの制御とプレフィックス ベースのマッチング操作が行えます。この接続点は通常、ネクストホップ通知（アップまたはダウン）イベントで実行するかどうか決定するために使用されます。

ネクストホップのトラッキングにより、BGP はBGP プレフィックスに直接影響を与えるRouting Information Base（RIB）内のルートの到達可能性をモニタできます。BGP ネクスト ホップ接続点でのルート ポリシーは、特定のプレフィックス向けに BGP に配信される通知を抑制するのに役立ちます。ルート ポリシーは RIB ルートに割り当てられます。通常、非 BGP ルート監視のため、ルート ポリシーはネクスト ホップ トラッキングとともに使用されます。

次の例に、プレフィックス 10.0.0.0 およびプレフィックス長 8 を持つスタティック ルートまたは接続ルートを監視するための、ルート ポリシーを使用したBGP ネクスト ホップ トラッキング機能の設定方法を示します。

```
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

clear-policy 接続点によって、clear bgp コマンドを使用するときに、さまざまな AS パス マッチング操作に基づいたより詳細な制御が行えます。この接続点は、通常、AS パス ベースのマッチング操作に基づいて BGP フラップ統計情報をクリアするかどうか判断するときに使用します。

次の例に、セット my-as-set 内でマッチングを行う 1 つ以上の正規表現がルートに関連付けられたASパスにマッチした場合にin演算子が真となるルート ポリシーの設定方法を示します。マッチした場合、clear コマンドは関連付けられたフラップ統計情報をクリアします。

```
as-path-set my-as-set
```

```
  ios-regex '_12$',
  ios-regex '_13$'
end-set

route-policy policy_a
  if as-path in my-as-set then
    pass
  else
    drop
  endif
end-policy

clear bgp ipv4 unicast flap-statistics route-policy policy_a
```

Debug

debug 接続点を使用して、より詳細なプレフィックス ベースのマッチング演算が行えます。この接続点、通常、ルートのプレフィックスに基づいてさまざまな BGP コマンドのデバッグ出力をフィルタリングするために使用されます。

次に、プレフィックス長が 8 のプレフィックス 20.0.0.0 だけをパスするルート ポリシーを設定する例を示します。デバッグ出力はそのプレフィックスに対してだけに表示されます。

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

  endif
end-policy

debug bgp update policy_b
```

BGP 属性と演算子

このテーブルでは、接続点ごとの BGP 属性と演算子をまとめます。

表 *2 : BGP* 属性と演算子

接続点	属性	一致	セット
aggregation	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	community	is-empty matches-any matches-every	set set additive delete in delete not in delete all
	destination	in	—
	extcommunity cost	—	set set additive
	local-preference	is、ge、le、eq	set
	med	is、eg、ge、le	setset +set -
	next-hop	in	set
	origin	is	set
	source	in	—
	suppress-route	—	suppress-route
	weight	—	set

接続点	属性	一致	セット
allocate-label	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	community	is-empty matches-any matches-every	—
	destination	in	—
	label	—	set
	local-preference	is、ge、le、eq	—
	med	is、eg、ge、le	—
	next-hop	in	—
	origin	is	—
	source	in	—
clear-policy	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—

接続点	属性	一致	セット
dampening	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	community	is-empty matches-any matches-every	—
	dampening	—/	set dampening
	destination	in	—
	local-preference	is、ge、le、eq	—
	med	is、eg、ge、le	—
	next-hop	in	—
	origin	is	—
	source	in	—
debug	destination	in	—
default originate	med	—	set set + set -
	rib-has-route	in	—

接続点	属性	一致	セット
neighbor-in	as-path	in is-local length NA neighbor-is originates-from passes-through unique-length	prepend prepend most-recent remove as-path private-as replace
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	communitycommunity with ‘peeras’	is-empty matches-any matches-every	set set additive delete-in delete-not-in delete-all
	destination	in	—
	extcommunity cost	—	set set additive
	extcommunity rt	is-empty matches-any matches-every matches-within	set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every matches-within	—
	local-preference	is、ge、le、eq	set
	med	is、eg、ge、le	set set + set -

接続点	属性	一致	セット
	next-hop	in	set set peer address
	origin	is	set
	route-aggregated	route-aggregated	NA
	source	in	—
	weight	—	set

接続点	属性	一致	セット
neighbor-out	as-path	in is-local length — neighbor-is originates-from passes-through unique-length	prepend prepend most-recent remove as-path private-as replace
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	communitycommunity with ‘peeras’	is-empty matches-any matches-every	set set additive delete-in delete-not-in delete-all
	destination	in	—
	extcommunity cost	—	set set additive
	extcommunity rt	is-empty matches-any matches-every matches-within	set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every matches-within	—
	local-preference	is、ge、le、eq	set
	med	is、eg、ge、le	

接続点	属性	一致	セット
			set set + set - set max-unreachable set igp-cost
	next-hop	in	set set self
	origin	is	set
	path-type	is	—
	rd	in	—
	route-aggregated	route-aggregated	—
	source	in	—
	unsuppress-route	—	unsuppress-route
	vpn-distinguisher	—	set
neighbor-orf	orf-prefix	in	n/a

接続点	属性	一致	セット
network	as-path	—	prepend
	community	—	set set additive delete-in delete-not-in delete-all
	destination	in	—
	extcommunity cost	—	set set additive
	mpls-label	route-has-label	—
	local-preference	—	set
	med	—	set set+ set-
	next-hop	in	set
	origin	—	set
	route-type	is	—
	tag	is、ge、le、eq	—
	weight	—	set
next-hop	destination	in	—
	protocol	is、in	—
	source	in	—

接続点	属性	一致	セット
redistribute	as-path	—	prepend
	community	—	set set additive delete in delete not in delete all
	destination	in	—
	extcommunity cost	—	setset additive
	local-preference	—	set
	med	—	set set+ set-
	next-hop	in	set
	origin	—	set
	mpls-label	route-has-label	—
	route-type	is	—
	tag	is、eq、ge、le	—
	weight	—	set
retain-rt	extcommunity rt	is-empty matches-any matches-every matches-within	—

接続点	属性	一致	セット
show	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	community	is-empty matches-any matches-every	—
	destination	in	—
	extcommunity rt	is-empty matches-any matches-every matches-within	—
	extcommunity soo	is-empty matches-any matches-every matches-within	—
	med	is、eg、ge、le	—
	next-hop	in	—
	origin	is	—
	source	in	—

一部の BGP ルート属性は、さまざまな理由のため一部の BGP 接続点からアクセスできません。たとえば、set med igp-cost only コマンドは、設定された IGP コストがあり、ソース値を示す場合に意味があります。

次の表では、どの操作が有効であるか、およびどの場合に有効であるかをまとめます。

表 *3 :* 接続点により制限された *BGP* 動作

コマンド	インポート	エクスポート	集約	再配布
prepend as-path most-recent	eBGP のみ	eBGP のみ	該当なし	該当なし
replace as-path	eBGP のみ	eBGP のみ	該当なし	該当なし
set med igp-cost	禁止	eBGP のみ	禁止	禁止
set weight	該当なし	禁止	該当なし	該当なし
suppress	禁止	禁止	該当なし	禁止

Default-Information Originate

default-information originate 接続点を使用して、条件に応じてデフォルトのルート 0.0.0.0/0 を OSPF リンクステート データベースに挿入できます。これはアタッチされたポリシーの評価によって実行されます。ローカル RIB の任意のルートがポリシーをパスすると、デフォルトのルートがリンクステート データベースに挿入されます。

次の例では、10.0.0.0/8 ge 8 le 25 に一致するルートが RIB に存在する場合に、デフォルトのルートを生成する方法を示します。

```
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 ポリシー接続点

この項では、それぞれの OSPF ポリシー接続点について説明し、OSPF 属性と演算子の概要を示します。

Redistribute

OSPF 内の redistribute 接続点は、他のルーティング プロトコル ソースから OSPF リンクステートデータベースにルートを挿入します。各プロトコルからインポートするルートを選択するこ

とで実行されます。その後、コストとメトリック タイプのOSPFパラメータを設定します。ポリシーは、set metric-type または set ospf-metric コマンドを使用して OSPF へのルート挿入を制御できます。

次の例に、ポリシー OSPF-redist を使用して IS-IS instance_10 から OSPF インスタンス 1 にルートを再配布する方法を示します。ポリシーは再配布されたすべてのルートでメトリック タイプを type-2 に設定します。タグ 10 を持つ IS-IS ルートはコストが 100 に設定され、タグ 20 を持つ IS-IS ルートは OSPF コストが 200 に設定されます。タグが 10 と 20 のいずれでもない IS-IS ルートは OSPF リンクステート データベースに再配布されません。

```
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

OSPF 内の area-in 接続点では、受信 OSPF タイプ 3 サマリー リンク ステート アドバタイズメント（LSA）をフィルタリングできます。この接続点ではプレフィックス ベースのマッチングが行えるため、より詳細なタイプ 3 サマリー LSA のフィルタリングが実現します。

次の例では、OSPF サマリー LSA のプレフィックスの設定方法を示します。プレフィックスが 10 .105.3.0/24、10 .105.7.0/24、10 .105.13.0/24 のいずれかに一致する場合、受け入れられます。プレフィックスが 10 .106.3.0/24、10 .106.7.0/24、10 .106.13.0/24 のいずれかに一致する場合、ドロップされます。

```
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

OSPF 内の area-in 接続点では、発信 OSPF タイプ 3 サマリー LSA をフィルタリングできます。この接続点ではプレフィックス ベースのマッチングが行えるため、より詳細なタイプ 3 サマリー LSA のフィルタリングが実現します。

次の例では、OSPF サマリー LSA のプレフィックスの設定方法を示します。プレフィックスが 10 .105.3.0/24、10 .105.7.0/24、10 .105.13.0/24 のいずれかに一致する場合、通知されます。プレフィックスが 10 .105.3.0/24、10 .105.7.0/24、10 .105.13.0/24 のいずれかに一致する場合、ドロップされ、通知はされません。

```
  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 属性と演算子

この表では接続点ごとの OSPF 属性と演算子をまとめます。

表 *4 : OSPF* 属性と演算子

接続点	属性	一致	セット
default-information originate	ospf-metric	—	set
	metric-type	—	set
	tag	—	set
	rib-has-route	in	—

接続点	属性	一致	セット
redistribute	destination	in	—
	metric-type	—	set
	ospf-metric	—	set
	next-hop	in	—
	mpls-label	route-has-label	—
	rib-metric	is、le、ge、eq	n/a
	route-type	is	—
	tag	is、eq、ge、le	set
area-in	destination	in	—
area-out	destination	in	—
spf-prefix-priority	destination	in	n/a
	spf-priority	n/a	set
	tag	is、le、ge、eq	n/a

Distribute-list in

OSPF 内の distribute-list in 接続点では、ルート ポリシーを使用して、OSPF プレフィックスをフィルタリングできます。distribute-list in ルート ポリシーは、OSPF インスタンス、エリア、およびインターフェイス レベルで設定できます。distribute-list in コマンドで使用されるルート ポリシーは、match ステートメント、「destination」および「rib-metric」をサポートします。「set」コマンドはルート ポリシーではサポートされません。

次は、「distribute-list in」の有効なルート ポリシーの例です。

```
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 ポリシー接続点

この項では、それぞれの OSPFv3 ポリシー接続点について説明し、OSPFv3 属性と演算子の概要を示します。

Redistribute

OSPFv3 内の redistribute 接続点は、他のルーティング プロトコル ソースから OSPFv3 リンクステート データベースにルートを挿入します。各プロトコルからインポートするルート タイプを選択することで実行されます。その後、コストとメトリック タイプの OSPFv3 パラメータを設定します。ポリシーは、metric type コマンドを使用して OSPFv3 へのルート挿入を制御できます。

次の例に、ポリシー OSPFv3-redist を使用して BGP インスタンス 15 から OSPF インスタンス 1 にルートを再配布する方法を示します。ポリシーは再配布されたすべてのルートでメトリック タイプを type-2 に設定します。タグ 10 を持つ BGP ルートはコストが 100 に設定され、タグ 20 を持つ BGP ルートは OSPFv3 コストが 200 に設定されます。タグが 10 と 20 のいずれでもない BGP ルートは OSPFv3 リンクステート データベースに再配布されません。

```
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 属性と演算子

この表では、接続点ごとの OSPFv3 属性と演算子をまとめます。

表 *5 : OSPFv3* 属性と演算子

接続点	属性	一致	セット
default-information originate	ospf-metric	—	set
	metric-type	—	set
	tag	—	set
	rib-has-route	in	—
redistribute	destination	in	—
	ospf-metric	—	set
	metric-type	—	set
	route-type	is	—
	tag	is、eq、ge、le	—

IS-IS ポリシー接続点

この項では、それぞれの IS-IS ポリシー接続点について説明し、IS-IS 属性と演算子の概要を示します。

Default-Information Originate

IS-IS 内の default-information originate 接続点を使用して、デフォルトのルート 0.0.0.0/0 を条件に応じて IS-IS ルート データベースに挿入できます。

次の例では、10.0.0.0/8 ge 8 le 25 に一致するルートが RIB に存在する場合に、IPv4 ユニキャストのデフォルト ルートを生成する方法を示します。IS-IS データベースに挿入されるデフォルト ルート上で、IS-IS ルートのコストは 100 に、レベルは level-1-2 に設定されます。

```
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

IS-IS 内の inter-area-propagate 接続点を使用して、プレフィックスをあるレベルから同じ IS-IS インスタンス内の別のレベルへと条件に応じてプロパゲートできます。

次の例に、プレフィックスのいずれかが 10.0.0.0/8 ge 8 le 25 に一致した場合に、プレフィックスをレベル 1 LSP からレベル 2 LSP へリークさせる方法を示します。

```
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

IS-IS 内の inter-area-propagate 接続点を使用して、プレフィックスをあるレベルから同じ IS-IS インスタンス内の別のレベルへと条件に応じてプロパゲートできます。

次の例に、プレフィックスのいずれかが 10.0.0.0/8 ge 8 le 25 に一致した場合に、プレフィックスをレベル 1 LSP からレベル 2 LSP へリークさせる方法を示します。

```
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
    .
```

Default-Accept-In

default-accept-in 接続点を使用すると、付加されたポリシーの評価によって EIGRP ルートの条件付きデフォルト フラグの設定とリセットが行えます。

次の例に、10.0.0.0/8 にマッチするルートと最大 10.0.0.0/25 までの長いプレフィックスすべてに条件付きデフォルト フラグを設定するポリシーを示します。

```
route-policy eigrp-cd-policy-in
  if destination in (10.0.0.0/8 ge 8 le 25) then
    pass
  endif
end-policy
!
router eigrp 100
  address-family ipv4
```

```
default-information allowed in route-policy eigrp-cd-policy-in
.
.
.
```

ルーティング ポリシーの非破壊編集

ルーティング ポリシーの非破壊編集によって、ルーティング ポリシー コンフィギュレーション モードでのデフォルトの終了動作が設定を中断するように変更されます。

デフォルトの **exit** コマンドは、end-policy、end-set、または end-if として機能します。**exit** コマンドがルートポリシーコンフィギュレーションモードで実行される場合は、変更が適用され、設定が更新されます。これによって、既存のポリシーが破壊されます。**rpl set-exit-as-abort** コマンドを使用すると、ルート ポリシー コンフィギュレーション モードで **exit** コマンドのデフォルトの動作を上書きできます。

アタッチされたポリシーの変更

使用中のポリシーを変更する必要が生じる場合もあります。従来のコンフィギュレーションモデルでのポリシーの変更では、いったんポリシーを完全に削除してから再入力再入力していました。しかし、このモデルではポリシーがアタッチされずデフォルトのアクションが使用される時間帯が生じてしまうため、不一致が発生する可能性がありました。この時間帯をなくすためには、接続点で使用中のポリシーを再指定することで変更を行います。使用中の変更対象ポリシーを、接続点にどのポリシーも適用されない時間帯をつくらずに変更できます。

（注） 接続点で使用中のルートポリシーまたはセットは削除できません。削除によって未定義の参照が生じるからです。接続点で使用中のルートポリシーまたはセットを削除しようとした場合、ユーザにはエラーメッセージが表示されます。

アタッチされないポリシーの変更

ポリシーは、接続点にアタッチされていないのであれば、存在していないセットやポリシーを参照することが許可されます。まだ定義されていない参照セットまたはポリシーブロックの設定を構築でき、その後、それらの定義されていないポリシーおよびセットに入力できます。この設定を構築する方法によって、ポリシー定義の柔軟性がより高まります。参照するポリシーの各部分は、ポリシーを定義しているときに設定に存在する必要はありません。このため、ポリシー sample2 が存在しない場合でも、apply ステートメントよってポリシー sample2 を参照するポリシー sample1 を定義できます。同様に、存在しないセットを参照するポリシー ステートメントを入力できます。

ただし、参照されているすべてのポリシーとセットの存在は、ポリシーが付加されたときに適用されます。このため、ステートメント **neighbor 1.2.3.4 address-family ipv4 unicast policy sample1 in** を使用してインバウンド BGP ポリシーで未定義ポリシー sample2 を参照するポリ

シー sample1 をアタッチしようとすると、ポリシー sample2 が存在しないために設定が拒否されます。

ルーティング ポリシー設定要素の編集

RPL は、行ではなくステートメントをベースにしています。つまり、CLI からのポリシー ステートメントをはさむ begin と end のペアの内部では、改行はセパレータでしかなく、スペース文字も同様です。

CLI によってルート ポリシー ステートメントの入力や削除が行えます。RPL では、begin と end の間にはさまれたポリシーの内容をテキスト エディタで編集する手順が準備されています。ソフトウェアでは、RPL ポリシーの編集に次のテキスト エディタを利用できます。

- Nano（デフォルト）
- Emacs
- Vim

Emacs エディタを使用したルーティング ポリシー設定要素の編集

Emacs エディタを使用してルーティング ポリシーの内容を編集するには、XR EXEC モードで次の CLI コマンドを使用します。

```
edit
  route-policy
  name
  emacs
```

ルート ポリシーのコピーが一時ファイルにコピーされ、エディタが起動します。編集後に、Ctrl+X および Ctrl+S のキーストロークを使用して編集バッファを保存します。エディタを保存して終了するには、Ctrl+X および Ctrl+C のキーストロークを使用します。エディタを終了すると、バッファがコミットされます。解析エラーがなければ、コンフィギュレーションがコミットされます。

```
RP/0/RP0/cpu 0: 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
```

解析エラーがある場合は、編集を続行するかどうかを尋ねられます。

```
RP/0/RP0/cpu 0: 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]:
```

yes と答えると、エディタは、中断した場所からテキストバッファを続行します。**no** と答えると、実行コンフィギュレーションは変更されず、編集セッションは終了します。

Vim エディタを使用したルーティング ポリシー設定要素の編集

Vim（Vi IMproved）を使用したルーティング ポリシーの要素の編集は、保存や終了のキーストロークといった一部機能の違いを除き、Emacs での編集に似ています。現在のファイルに書き込んで終了するには、**:wq**、**:x**、または **ZZ** のキーストロークを使用します。終了して確認するには、**:q** キーストロークを使用します。終了して変更を廃棄するには、**:q!** キーストロークを使用します。

Vim の詳細なオンライン マニュアルは、次の URL から参照できます。http://www.vim.org/

CLI を使用したルーティング ポリシー設定要素の編集

CLI を使用して、ルート ポリシー ステートメントの入力や削除が行えます。ポリシー コンフィギュレーション ブロックは、**end-policy** や **end-set** といった適用可能なコマンドの入力によって完了できます。または、CLI インタープリタでは exit コマンドでポリシー コンフィギュレーション ブロックを完了させることも可能です。現在のポリシー コンフィギュレーションを廃棄して モードに戻るには、**abort** コマンドを使用します。

Nano エディタを使用したルーティング ポリシー設定要素の編集

Nanoエディタを使用してルーティングポリシーの内容を編集するには、XR EXECモードで次のCLIコマンドを使用します。

```
edit route-policy
  name
  nano
```

ルートポリシーのコピーが一時ファイルにコピーされ、エディタが起動します。編集後、Ctrl-Xを入力してファイルを保存し、エディタを終了します。利用可能なエディタのコマンドは画面上に表示されます。

Nano エディタの使用について詳しくは、次の URL を参照してください。
http://www.nano-editor.org/

ソフトウェアでは、Nano エディタの機能の一部がサポートされていません。

XML を使用したルーティング ポリシー設定要素の編集

RPLは、XMLを使用した設定要素の編集をサポートしています。XML経由で、既存のセットを置き換えることなくエントリの後方追加、前方追加、削除が行えます。

階層型ポリシー条件

階層型ポリシー条件機能は、他のルートポリシーの「if」ステートメント内のルートポリシーを指定する機能をイネーブルにします。この機能によって、ルートポリシーを階層的ポリシーに基づいたコンフィギュレーションに適用できます。

階層型ポリシー条件機能によって、ソフトウェアではさまざまなマッチング文に加え、さまざまな種類の Boolean 演算子とともに使用できる条件ポリシーの適用をサポートしています。

条件ポリシーの適用

適用条件ポリシーを利用すると、ルートポリシーを別のルートポリシーの「if」ステートメント内で使用できます。

Parent、*Child A*、および *Child B* ルート ポリシー コンフィギュレーションを検討します。

```
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
!
```

上記のシナリオでは、*Parent* ポリシーが実行されるたびに、ポリシー *Child A* および *Child B* の結果に基づいて、ポリシーの中の「if」条件の判断が選択されます。ポリシー *Parent* は、次に示すように、ポリシー *merged* に相当します。

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

条件の適用はパラメータとともに使用され、すべての接続点およびすべてのクライアントでサポートされます。条件の階層的適用は、カスケード レベル上で制約なく使用できます。

既存のルート ポリシー セマンティックは、この条件適用を含むように拡張できます。

```
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
```

単純な階層型ポリシーの pass/drop/done RPL ステートメントの動作

次の表は、単純な階層型ポリシーの **pass**/**drop**/**done** RPL ステートメントの動作、および単純な階層型ポリシーの考えられる **done** ステートメントの実行シーケンスを説明します。

単純な階層型ポリシーのあるルート ポリシー	考えられる **done** ステートメントの実行シーケンス	動作
pass	**pass** Continue_list	プレフィックスに「acceptable」としてマークを付け、continue_list ステートメントの実行を続けます。
drop	Stmts_list **drop**	**drop** ステートメントにヒットするとただちにルートを拒否し、ポリシー実行を停止します。
done	Stmts_list **done**	**done** ステートメントにヒットするとただちにルートを受け入れ、ポリシー実行を停止します。
pass それから **done**	**pass** Statement_list **done**	「accept route」のある **done** ステートメントでは、ただちに終了します。
drop それから **done**	**drop** Statement list **done**	これは、実行時点では無効なシナリオです。ポリシーはステートメント リストまたは **done** ステートメントには進まずに、**drop** ステートメント自体で実行を終了します。プレフィックスは、拒否またはドロップされます。

階層型ポリシー条件の pass/drop/done RPL ステートメントの動作

次の項では、階層型ポリシー条件の **pass**/**drop**/**done** RPL ステートメントの動作、および階層型ポリシー条件の考えられる **done** ステートメントの実行シーケンスを説明します。

ポリシー実行の用語：「true-path」と、「false-path」、および「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
```

階層型ポリシー条件	考えられる **done** ステートメントの実行シーケンス	動作
pass	**pass** Continue_list	戻り値を「true」とし、同じポリシー条件内で実行を続けます。 「**pass**」の後にステートメントがない場合は、「true」を返します。
pass それから **done**	**pass** または **set** アクション ステートメント Stmt_list **done**	戻り値を「true」とし、**done** ステートメントまで実行を続けます。「true-path」になる適用ポリシー条件に「true」を返します。
done	**pass** または **set** 操作のない Stmt_list DONE	「false」を返します。条件は「false-path」になります。
drop	Stmt_list **drop** Stmt_list	プレフィックスはドロップされるか拒否されます。

ネストされたワイルドカード適用ポリシー

ルーティングポリシー言語（RPL）の階層構造では、ポリシーが異なるポリシーを参照できます。参照されている、または呼び出されているポリシーは、子ポリシーと呼ばれます。別のポリシーを参照するまたは呼び出すポリシーは、親ポリシーと呼ばれます。呼び出すポリシーまたは親ポリシーは、BGP ネイバーの共通セットとの接続用の複数の子ポリシーをネストできます。ネストされたワイルドカード適用ポリシーでは、適用のネスティングに基づくワイルドカード（*）が可能です。ワイルドカード操作では、ルータ上に定義される、特定の定義済み英数字セットを含むポリシーすべてを呼び出す一般的な apply ステートメントを宣言することが許されています。

ワイルドカードは、apply ステートメントにポリシー名の最後にアスタリスク（*）を配置することによって指定されます。ワイルドカードポリシーに、パラメータを渡すことはサポートされていません。ワイルドカードは、適用ポリシーのその部分が任意の値と一致することを示します。

ネストされたワイルドカード適用ポリシーの例として、次のようなポリシー階層を考えてみます。

```
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
```

ここで、単一の親 apply ステートメント（apply service_policy_customer*）が、指定された文字列「service_policy_customer」を含むすべての子ポリシーを呼び出して（継承して）います。それぞれの子ポリシーをグローバルに定義されると、親ポリシーは動的にポリシー名に基づいて子ポリシーをネストします。親ポリシーを設定すると、各子ポリシーをオンデマンドで継承します。親ポリシーと子ポリシーの間に、ワイルドカード一致ステートメント以上の直接的な関連付けはありません。

VRF インポート ポリシーの強化

VRF RPL ベースのインポート ポリシー機能では、ルートターゲット（RT）およびポリシー内で指定された他の基準を照合することによって、インポートルートポリシーだけに基づいてインポート操作を実行する機能が提供されます。グローバル VRF アドレス ファミリ コンフィギュレーション モードでインポート RT を明示的に設定する必要はありません。インポート RT およびインポート ルートポリシーがすでに定義されている場合、ルートはインポート RT で設定された RT からインポートされ、インポート ルートポリシーで接続されたルートポリシーに従います。

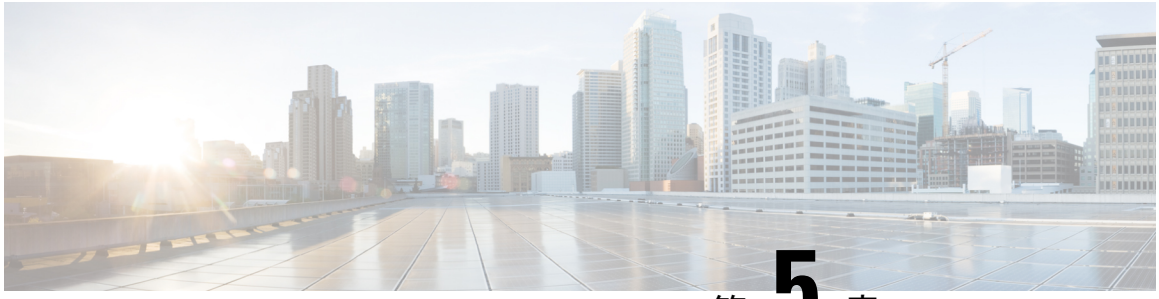
この機能を有効にするには、VPN アドレスファミリ コンフィギュレーション モードの VRF サブモード下で **source rt import-policy** コマンドを使用します。

集約されたルートの照合

集約されたルートの照合機能は、集約されていないルートから BGP 集約ルートを照合するのに役立ちます。BGPは、ネイバーに更新を送信する前に、ルートのグループを1つのプレフィックスに集約できます。集約されたルートの照合機能を使用して、ルート ポリシーはこの集約されたルートを他のルートから切り離します。

着信ポリシーでのプライベート AS の削除

BGPは、ネイバーにパケットを送信する前に、自身の as-path を付加します。パケットが複数の iBGP ネイバーを通過すると、as-path 構造には、多くのプライベート自律システム（AS）が追加されます。着信ポリシーでのプライベート AS の削除では、RPL route-policy を使用してそれらのプライベート自律システムを削除する機能が与えられます。 **remove as-path private-as** コマンドを使用すると、AS 番号が 64512 ～ 65535 の自律システム（AS）が削除されます。



第 5 章

スタティック ルートの実装

スタティック ルートは、指定のパスを通るように発信元と宛先の間でパケットを移動させるユーザ定義のルートです。スタティック ルートは、ソフトウェアが特定の宛先へのルートを確立できない場合に重要になることがあります。また、ルーティングできないすべてのパケットを送るラスト リゾート ゲートウェイを指定する場合にも役立ちます。

スタティック ルートの参照 （205 ページ） では、スタティック ルートに関する追加の概念情報を提供します。

このモジュールでは、スタティック ルートの実装方法について説明します。

- スタティック ルートの実装に関する制約事項 （197 ページ）
- スタティック ルートの設定 （198 ページ）
- フローティング スタティック ルート （199 ページ）
- PE-CE ルータ間でのスタティック ルートの設定 （200 ページ）
- IPv4 マルチキャスト スタティック ルート （202 ページ）
- デフォルト VRF （204 ページ）
- スタティック ルートの参照 （205 ページ）

スタティック ルートの実装に関する制約事項

次の制約事項は、スタティック ルートの実装時に適用されます。

- ローカル サブネットの一部である間接ネクスト ホップへのスタティック ルーティング（RIB によって学習されたプレフィックス。AIB ではより具体的である可能性がある）では、出力インターフェイスを示すグローバル テーブルで、スタティック ルートをネクスト ホップとして設定する必要があります。転送のドロップを避けるには、ネクスト ホップ IP アドレスを示すグローバル テーブルで、スタティック ルートがネクスト ホップになるように設定します。

- 通常、ルートは、グローバル テーブルの AIB から学習され、FIB にインストールされます。ただし、この動作はリークされたプレフィックスには繰り返されません。これは、転送動作の不整合の原因となることがあります。

スタティック ルートの設定

スタティック ルートは、すべてユーザが設定であり、ネクスト ホップ インターフェイス、ネクストホップIPアドレス、またはその両方を指示できます。ソフトウェアでは、インターフェイスが指定された場合、そのインターフェイスが到達可能であれば、スタティック ルートがルーティング情報ベース（RIB）にインストールされます。インターフェイスが指定されていない場合、ネクストホップ アドレスが到達可能であれば、そのルートはインストールされます。このコンフィギュレーションの唯一の例外は、スタティック ルートに permanent 属性が設定されている場合です。このときは到達可能性にかかわらず RIB にインストールされます。

ここでは、スタティック ルートを設定する方法について説明します。

手順

ステップ 1 **configure**

ステップ 2 **router static**

例：

```
RP/0/RP0/cpu 0: router(config)# router static
```

スタティック ルート コンフィギュレーション モードを開始します。

ステップ 3 **vrf** *vrf-name*

例：

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

（任意）VRF コンフィギュレーション モードを開始します。

VRF が指定されていない場合、スタティック ルートはデフォルトの VRF で設定されます。

ステップ 4 **address-family** { **ipv4** | **ipv6** } { **unicast** | **multicast** }

例：

```
RP/0/RP0/cpu 0: router(config-static-vrf)# address family ipv4 unicast
```

アドレス ファミリ モードを開始します。

ステップ 5 *prefix mask* [**vrf** *vrf-name*] { *ip-address* | *interface-type* *interface-instance* } [*distance*] [**description** *text*] [**tag** *tag*] [**permanent**]

例：

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

アドミニストレーティブ ディスタンス 110 を設定します。

- 次に、アドミニストレーティブディスタンスが110より小さいダイナミック情報が使用できない場合、172.20.16.6 のネクスト ホップを介してネットワーク 10.0.0.0 のパケットをルーティングする方法の例を示します。

ステップ 6 **commit**

デフォルトのスタティック ルートは、多くの場合、単純なルータ トポロジで使用されます。次の例では、アドミニストレーティブ ディスタンス 110 でルートが設定されます。

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

フローティング スタティック ルート

フローティング スタティック ルートは、設定されたルーティング プロトコルを介して学習されたダイナミック ルートのバックアップに使用されるスタティック ルートです。フローティング スタティック ルートには、バックアップしているルーティング プロトコルよりも大きなアドミニストレーティブ ディスタンスが設定されています。このため、ルーティング プロトコルを介して学習されたダイナミック ルートは、フローティング スタティック ルートよりも常に優先して使用されます。ルーティング プロトコルを介して学習されたダイナミック ルートが失われると、フローティング スタティック ルートが代わりに使用されます。

（注） デフォルトでは、スタティック ルートはダイナミック ルートよりアドミニストレーティブディスタンスが小さいため、スタティック ルートがダイナミック ルートに優先されます。

フローティング スタティック ルートの設定

ここでは、フローティング スタティック ルートを設定する方法について説明します。

手順

ステップ 1 **configure**

ステップ 2 **router static**

例：

```
RP/0/RP0/cpu 0: router(config)# router static
```

スタティック ルート コンフィギュレーション モードを開始します。

ステップ 3 **vrf** *vrf-name*

例：

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

（任意）VRF コンフィギュレーション モードを開始します。

VRF が指定されていない場合、スタティック ルートはデフォルトの VRF で設定されます。

ステップ 4 **address-family** { **ipv4** | **ipv6** } { **unicast** | **multicast** }

例：

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

アドレス ファミリ モードを開始します。

ステップ 5 *prefix mask* [**vrf** *vrf-name*] { *ip-address* | *interface-type* *interface-instance* } [*distance*] [**description** *text*] [**tag** *tag*] [**permanent**]

例：

```
RP/0/RP0/cpu 0: router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
```

アドミニストレーティブ ディスタンス 201 を設定します。

ステップ 6 **commit**

フローティング スタティック ルートは、しばしば接続失敗時のバックアップ パスの準備として使用されます。次の例では、アドミニストレーティブ ディスタンス 201 でルートが設定されます。

```
configure
 router static
  address-family ipv6 unicast
  2001:0DB8::/32 2001:0DB8:3000::1 201
  end
```

PE-CE ルータ間でのスタティック ルートの設定

このタスクでは、PE-CE ルータ間でのスタティック ルーティングの設定方法について説明します。

（注） 6VPE（IPv6 VPN Provider Edge）では、VRF フォールバックはサポートされていません。

手順

ステップ 1 **configure**

ステップ 2 **router static**

例：

```
RP/0/RP0/cpu 0: router(config)# router static
```

スタティック ルート コンフィギュレーション モードを開始します。

ステップ 3 **vrf** *vrf-name*

例：

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

（任意）VRF コンフィギュレーション モードを開始します。

VRF が指定されていない場合、スタティック ルートはデフォルトの VRF で設定されます。

ステップ 4 **address-family** { **ipv4** | **ipv6** } { **unicast** | **multicast** }

例：

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

アドレス ファミリ モードを開始します。

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

例：

```
RP/0/RP0/cpu 0: router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
```

アドミニストレーティブ ディスタンス 201 を設定します。

ステップ 6 **commit**

次の例では、PE ルータと CE ルータ間のスタティック ルートが設定され、VRF がスタティック ルートに関連付けられます。

```
configure
 router static
  vrf vrf_A
```

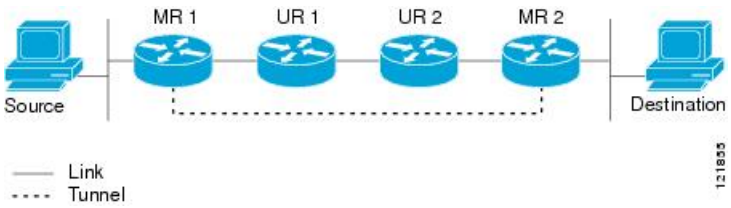
```
address-family ipv4 unicast
0.0.0.0/0 2.6.0.2 120
end
```

IPv4 マルチキャスト スタティック ルート

IP マルチキャスト スタティック ルート（mroute）を使用すると、マルチキャスト パスをユニキャスト パスから分岐させることができます。Protocol Independent Multicast（PIM）を使用する場合、ルータは、ユニキャストパケットを発信元に返送するときと同じインターフェイスでパケットを受信することを想定しています。この想定は、マルチキャストとユニキャストのトポロジが一致している場合に有益です。しかし、ユニキャストパケットに使用するパスとは別のパスをマルチキャストに使用することも考えられます。

別々のユニキャスト パスおよびマルチキャスト パス を使用する最も一般的な理由はトンネリングです。発信元と宛先の間のパスがマルチキャスト ルーティングをサポートしていない場合は、その間に GRE トンネルを持つ 2 つのルータを設定することがソリューションです。下の図では、各ユニキャスト ルータ（UR）はユニキャスト パケットのみをサポートしています。各マルチキャスト ルータ（MR）はマルチキャスト パケットをサポートします。

図 *5:* マルチキャスト パケットのトンネル



図では、送信元は MR 1 および MR 2 を使用してマルチキャスト パケットを宛先に伝送します。MR 2 は、トンネルを経由して送信元に到達できることを予測している場合だけ、マルチキャスト パケットを受け取ります。この状況が真である場合、宛先がユニキャスト パケットを送信元に送信すると、MR2 はそれらをトンネルを介して送信します。MR2 がトンネルを介して送信元に到達できるかどうかの確認は、リバース パス フォワーディング（RPF）のチェックであり、マルチキャストパケットが到着するインターフェイスが送信元に戻るユニキャスト パスではない場合は、スタティック mroute によってそのチェックが正常に行われます。トンネルを介したパケットの送信は、UR2、UR1、および MR1 を介してネイティブに送信するよりも遅くなる可能性があります。

マルチキャスト スタティック ルートを使用すると、スタティック マルチキャスト ソースを設定することにより、上図の構成を使用できます。システムは、ユニキャスト ルーティング テーブルの代わりに設定情報を使用してトラフィックをルーティングします。したがって、ユニキャスト パケットにトンネルを使用させることなく、マルチキャスト パケットがトンネルを使用できます。スタティック mroute は設定されているルータにローカルであり、他のルータにアドバタイズされたり再分配されたりすることはありません。

マルチキャスト スタティック ルートの設定

次に、IPv4 および IPv6 のアドレス ファミリ コンフィギュレーション モードで複数のスタティック ルートを設定する例を示します。

```
/* Enables a static routing process */
Router(config)# router static

/* Configures the IPv4 address-family for the unicast topology with a destination prefix.
 */
Router(config-static)# address-family ipv4 unicast
Router(config-static-afi)# 10.1.1.0/24 198.51.100.1
Router(config-static-afi)# 223.255.254.254/32 203.0.113.1
Router(config-static-afi)# exit

/* Configures the IPv4 address-family for the multicast topology with a destination
prefix. */
Router(config-static)# address-family ipv4 multicast
Router(config-static-afi)# 198.51.100.20/32 209.165.201.0
Router(config-static-afi)# 192.0.2.10/32 209.165.201.0
Router(config-static-afi)# exit

/* Enable the address family IPv4 and IPv6 multicast on the next hop interface. */
Router(config)# interface TenGigE 0/0/0/0
Router(config-if)# address-family ipv4 multicast
Router(config-if)# address-family ipv6 multicast
```

実行コンフィギュレーション

```
router static
 address-family ipv4 unicast
  10.1.1.0/24 198.51.100.1
  223.255.254.254/32 203.0.113.1
  !
address-family ipv4 multicast
  198.51.100.20/32 209.165.201.0
  192.0.2.10/32 209.165.201.0
  !
interface TenGigE 0/0/0/0
  address-family ipv4 multicast
  address-family ipv6 multicast
```

確認

IPv4 マルチキャストのルートを確認します。

```
show route ipv4 multicast

Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
        i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
        ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
        U - per-user static route, o - ODR, L - local, G  - DAGR, l - LISP
        A - access/subscriber, a - Application route
        M - mobile route, r - RPL, t - Traffic Engineering, (!) - FRR Backup path

Gateway of last resort is 10.1.1.20 to network 0.0.0.0
```

```
i*L1 0.0.0.0/0 [115/10] via 10.1.1.20, 00:41:12, TenGigE0/0/0/6
C    10.1.1.0/24 is directly connected, 00:41:12, TenGigE0/0/0/0
L    10.1.1.10/32 is directly connected, 00:41:12, TenGigE0/0/0/0
S    172.16.2.10/32 [1/0] via 198.51.100.20, 00:41:12
i L1 172.16.3.1/32 [115/20] via 198.51.100.20, 00:41:12, TenGigE0/0/0/12
i L1 192.0.2.1/24 [115/20] via 198.51.100.20, 00:41:12, TenGigE0/0/0/1
```

デフォルト VRF

スタティック ルートは常に VPN ルーティング/転送（VRF）インスタンスに関連付けられます。VRF には、デフォルト VRF または指定の VRF を設定できます。**vrf** *vrf-name* コマンドを使用して VRF を指定することで、指定の VRF の VRF コンフィギュレーション モードに入り、スタティック ルートを設定できます。VRF が指定されない場合、デフォルトの VRF スタティック ルートが設定されます。

（注） IPv4 または IPv6 スタティック VRF ルートは、デフォルト VRF 用に設定されたスタティック ルートと同じです。IPv4 および IPV6 アドレス ファミリがそれぞれの VRF でサポートされます。

スタティック ルートを使用した VRF の関連付け

ここでは、VRF をスタティック ルートと関連付ける方法について説明します。

手順

ステップ 1 **configure**

ステップ 2 **router static**

例：

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

スタティック ルート コンフィギュレーション モードを開始します。

ステップ 3 **vrf** *vrf-name*

例：

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

VRF コンフィギュレーション モードを開始します。

ステップ 4 **address-family** { **ipv4** | **ipv6** } { **unicast** | **multicast** }

例：

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

アドレス ファミリ モードを開始します。

ステップ 5 *prefix mask* [**vrf** *vrf-name*] {**next-hop** *ip-address* | *interface-name* } {*path-id* } [*distance*] [**description** *text*] [**tag** *tag*] [**permanent**]

例：

```
RP/0/RP0/cpu 0: router(config-static-vrf-afi)# 2001:0DB8::/32 2001:0DB8:3000::1 201
```

アドミニストレーティブ ディスタンス 201 を設定します。

ステップ 6 **commit**

スタティック ルートの参照

次のトピックでは、スタティック ルートに関する追加の概念情報について説明します。

- スタティック ルート機能の概要 （205 ページ）
- デフォルトのアドミニストレーティブ ディスタンス （206 ページ）
- 直接接続されたルート （206 ページ）
- フローティング スタティック ルート （199 ページ）
- 完全指定のスタティック ルート （207 ページ）
- 再帰スタティック ルート （207 ページ）

スタティック ルート機能の概要

ネットワーキング デバイスでは、手動で設定したルート情報、またはルーティング プロトコルを使用してダイナミックに学習したルート情報を使用して、パケットを転送します。スタティック ルートは、手動で設定され、2 つのネットワーク デバイス間の明示パスを定義します。ダイナミック ルーティング プロトコルとは異なり、スタティック ルートは動的に更新されず、ネットワーク トポロジが変更された場合は手動で再設定する必要があります。スタティック ルートを使用する利点は、セキュリティが高まり、リソースが効率化されることです。スタティック ルートでは、ダイナミック ルーティング プロトコルよりも少ない帯域幅を使用し、ルートの計算および通信に CPU サイクルが使用されません。スタティック ルートを使用する場合の主なデメリットは、ネットワーク トポロジが変更された場合に自動的に再設定されないことです。

スタティック ルートはダイナミック ルーティング プロトコルに再配布できますが、ダイナミック ルーティング プロトコルによって生成されたルートは、スタティック ルーティング テーブルに再配布できません。スタティック ルートを使用するルーティング ループの設定を回避するアルゴリズムはありません。

スタティック ルートは、外部ネットワークへのパスが1つしかない小規模ネットワークでは有用です。また、大規模ネットワークの場合は、より厳格な制御が必要な、他のネットワークへの特定のタイプのトラフィックやリンクにセキュリティを提供します。一般に、大半のネットワークでは、ダイナミック ルーティング プロトコルを使用してネットワーキング デバイス間の通信を行いますが、特殊なケース用として1つまたは2つのスタティック ルートを設定している場合があります。

デフォルトのアドミニストレーティブ ディスタンス

スタティック ルートのデフォルトのアドミニストレーティブ ディスタンスは 1 です。小さい数値は、優先ルートを示します。デフォルトでは、スタティック ルートは、ルーティング プロトコルで学習したルートよりも優先されます。したがって、ダイナミック ルートでスタティック ルートを上書きさせる場合、スタティック ルートとともにアドミニストレーティブ ディスタンスを設定できます。たとえば、Open Shortest Path First（OSPF）プロトコルで追加される、アドミニストレーティブ ディスタンスが 120 のルートを設定できます。OSPF ダイナミック ルートで上書きされるスタティック ルートにするには、120 よりも大きいアドミニストレーティブ ディスタンスを指定します。

直接接続されたルート

ルーティング テーブルは、インターフェイスを指すスタティック ルートを「直接接続されている」と見なします。直接接続されたネットワークは、対応する interface コマンドがそのプロトコルのルータ設定のスタンザに含まれている場合、IGP ルーティング プロトコルによってアドバタイズされます。

直接接続されたスタティック ルートでは、出力インターフェイスだけが指定されます。宛先は、出力インターフェイスに直接接続されていると想定されるため、パケットの宛先はネクスト ホップ アドレスとして使用されます。次の例に、アドレス プレフィックス 2001:0DB8::/32 を持つ宛先すべてをインターフェイス TenGigE 0/0/0/0 経由で直接到達可能と指定する方法を示します。

```
RP/0/RP0/cpu 0: router(config)# router static
RP/0/RP0/cpu 0: router(config-static)# address-family ipv6 unicast
RP/0/RP0/cpu 0: router(config-static-afi)# 2001:0DB8::/32 TenGigE 0/0/0/0
```

直接接続されたスタティック ルートは、有効なインターフェイス（つまり、アップ状態にあり、かつ IPv4 または IPv6 がイネーブルになっているインターフェイス）を示している場合にかぎり、ルーティング テーブルに挿入される候補となります。

フローティング スタティック ルート

フローティング スタティック ルートは、設定されたルーティング プロトコルを介して学習されたダイナミック ルートのバックアップに使用されるスタティック ルートです。フローティング スタティック ルートには、バックアップしているルーティング プロトコルよりも大きなアドミニストレーティブ ディスタンスが設定されています。このため、ルーティング プロトコルを介して学習されたダイナミック ルートは、フローティング スタティック ルートよりも常に優先して使用されます。ルーティング プロトコルを介して学習されたダイナミック ルートが失われると、フローティング スタティック ルートが代わりに使用されます。

（注） デフォルトでは、スタティック ルートはダイナミック ルートよりアドミニストレーティブ ディスタンスが小さいため、スタティック ルートがダイナミック ルートに優先されます。

完全指定のスタティック ルート

完全指定のスタティック ルートでは、出力インターフェイスとネクスト ホップの両方が指定されています。この形式のスタティック ルートは、出力インターフェイスがマルチアクセス インターフェイスであり、ネクスト ホップを明示的に識別する必要がある場合に使用されます。ネクスト ホップは、指定した出力インターフェイスに直接接続されている必要があります。次の例に、完全指定のスタティック ルートの定義を示します。

```
RP/0/RP0/cpu 0: router(config)# router static
RP/0/RP0/cpu 0: router(config-static)# address-family ipv6 unicast
RP/0/RP0/cpu 0: router(config-static-afi)# 2001:0DB8::/32 TenGigE 0/0/0/0 2001:0DB8:3000::1
```

完全指定のルートが有効である（つまり、ルーティング テーブルに挿入される候補である）のは、指定された IPv4 または IPv6 インターフェイスがイネーブルで、アップ状態の場合です。

再帰スタティック ルート

再帰スタティック ルートでは、ネクスト ホップだけが指定されます。出力インターフェイスはネクスト ホップから取得されます。次の例に、アドレス プレフィックス 2001:0DB8::/32 を持つ宛先すべてをアドレス 2001:0DB8:3000::1 のホスト経由で到達可能と指定する方法を示します。

```
RP/0/RP0/cpu 0: router(config)# router static
RP/0/RP0/cpu 0: router(config-static)# address-family ipv6 unicast
RP/0/RP0/cpu 0: router(config-static-afi)# 2001:0DB8::/32 2001:0DB8:3000::1
```

再帰スタティック ルートが有効である（つまり、ルーティング テーブルに挿入される候補である）のは、指定したネクスト ホップが直接的または間接的に有効な出力インターフェイスに解決され、ルートが自己再帰型ではなく、再帰深度が IPv6 転送の最大再帰深度を超えていない場合だけです。

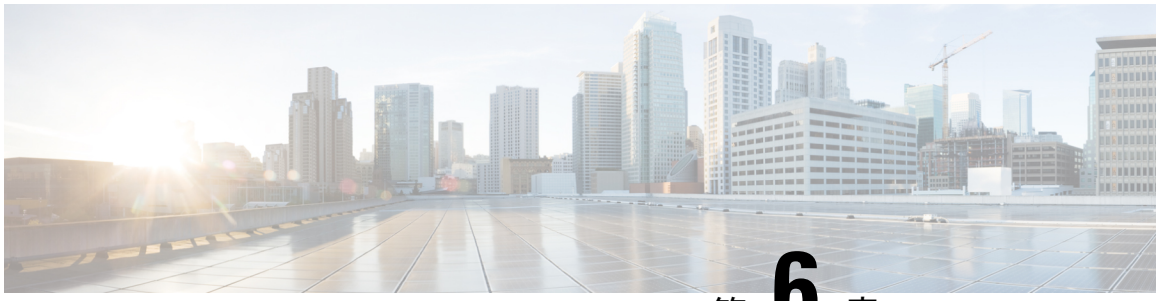
自身のネクストホップ解決に使用されるのがそのルート自身である場合、ルートは自己再帰します。スタティック ルートが自己再帰型になった場合、RIB は再帰ルートを除外するようスタティック ルートに通知を送ります。

BGP ルート 2001:0DB8:3000::0/16 のネクスト ホップが 2001:0DB8::0104 と仮定すると、次のスタティック ルートは IPv6 RIB に挿入されません。BGP ルート ネクストホップがそのスタティック ルートを介して解決される一方で、そのルートも BGP ルートを介して解決され、自己再帰型になるからです。

```
RP/0/RP0/cpu 0: router(config)# router static
RP/0/RP0/cpu 0: router(config-static)# address-family ipv6 unicast
RP/0/RP0/cpu 0: router(config-static-afi)# 001:0DB8::/32 2001:0DB8:3000::1
```

このスタティック ルートは、自己再帰型であるため、IPv6 ルーティング テーブルには挿入されません。スタティック ルートのネクスト ホップ 2001:0DB8:3000:1 は、自身が再帰ルートである（つまり、ネクスト ホップだけを指定する）BGP ルート 2001:0DB8:3000:0/16 を介して解決されます。BGP ルートのネクスト ホップ 2001:0DB8::0104 は、スタティック ルートを介して解決されます。したがって、スタティック ルートは、スタティック ルート自身のネクストホップを解決するために使用されることになります。

一般に、自己再帰型スタティック ルートの手動設定は禁止されていませんが、有用ではありません。ただし、ルーティング テーブルに挿入された再帰スタティック ルートが、ダイナミック ルーティング プロトコルを介して学習された、ネットワークでの何らかの一時的変更の結果として自己再帰になる場合があります。このような状況が発生すると、スタティック ルートが自己再帰になった事実が検出され、そのスタティック ルートはルーティング テーブルから削除されます（設定からは削除されません）。以降のネットワーク変更によって、スタティック ルートが自己再帰でなくなる場合があります。この場合、そのスタティック ルートはルーティング テーブルに再挿入されます。



第 6 章

BFD の実装

双方向フォワーディング検出（BFD）では、隣接する転送エンジン間のパスにおける障害を低オーバーヘッド、短期間で検出できます。BFD では、あらゆるメディアおよびあらゆるプロトコル レイヤでの障害検出に単一のメカニズムを使用でき、広範な検出時間とオーバーヘッドに対応できます。障害の迅速な検出が可能なため、リンクやネイバーの障害発生時にもただちに障害に対応することができます。

Cisco NCS 5500 ルータは、VRF コンテキストを使用した BFD をサポートしています。

- BFD の概要 （209 ページ）
- BFD の透過性 （216 ページ）

BFD の概要

双方向フォワーディング検出（BFD）では、隣接するルータ間のパスにおける障害を低オーバーヘッド、短期間で検出できます。BFD では、あらゆるメディアおよびあらゆるプロトコル レイヤでの障害検出に単一のメカニズムを使用でき、広範な検出時間とオーバーヘッドに対応できます。障害の迅速な検出が可能なため、リンクやネイバーの障害発生時にもただちに障害に対応することができます。

ルータは、VRF コンテキストを使用した BFD をサポートしています。

機能制限

BFD には、次の制約事項が適用されます。

- Cisco IOS XR ソフトウェアではデマンド モードはサポートされません。
- BFD エコー モードと暗号化はサポートされていません。
- IPv4 に対する BFD ハードウェア オフロードがサポートされています。
- スタティック、OSPF、BGP および IS-IS アプリケーションのみが BFD でサポートされています。
- IPv6 BFD ではスタティック ルートのみがサポートされています。

- BFD は IP コアでのみサポートされています。コアでは、ラベル配布プロトコル、セグメントルーティング、またはトラフィックエンジニアリングと共存することができません。
- バンドル機能を介した BFD では IETF モードのみがサポートされています。
- BFD の減衰拡張はサポートされていません。
- BoB と BLB の共存はサポートされていません。

ルータでの BFD の IPv6 チェックサム計算のイネーブル化およびディセーブル化

ルータ上で BFD の IPv6 チェックサム計算を設定するには、次の手順を実行します。

```
RP/0/RP0/CPU0:router(config)# bfd
RP/0/RP0/CPU0:router(config-bfd-if)# ipv6 checksum disable
RP/0/RP0/CPU0:router(config-bfd-if)# commit
```

ダイナミック ルーティング プロトコル下での BFD の設定またはスタティック ルートの使用

BFD ネイバーを確立するには、次の手順の少なくとも 1 つを実行し、ダイナミック ルーティング プロトコル下で BFD を設定するか、またはスタティック ルートを使用します。

インターフェイスでの OSPF への BFD の有効化

Open Shortest Path First（OSPF）での BFD を特定のインターフェイスで設定するには、次の手順を実行します。この方法の手順は、コマンドモードが異なる点を除き、IS-IS で BFD を設定する手順と共通です。

（注） インターフェイス単位での BFD の設定は、OSPF および IS-IS のみでサポートされます。

```
Router# configure

/* Enter OSPF configuration mode to configure the OSPF routing process. */
Router(config)# router ospf 0

/* Set the BFD minimum interval. The range is from 15 to 30000 milliseconds. */
Router(config-ospf)# bfd minimum-interval 6500

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

/* Configure an Open Shortest Path First (OSPF) area. */
Router(config-ospf)# area 0

/* Enter interface configuration mode. */
```

```
Router(config-ospf-ar)# interface gigabitEthernet 0/3/0/1

/* Enable BFD to detect failures in the path between adjacent forwarding engines. */
Router(config-ospf-ar-if)# bfd fast-detect
```

実行コンフィギュレーション

```
configure
  router ospf 0
  bfd minimum-interval 6500
  bfd multiplier 7
   area 0
    interface gigabitEthernet 0/3/0/1
     bfd fast-detect
```

確認

適切なインターフェイスで BFD がイネーブルになっていることを確認します。

```
Router(config-ospf-ar-if)# show run router ospf

 router ospf 0
 bfd minimum-interval 6500
 bfd multiplier 7
 area 0
 interface gigabitEthernet 0/3/0/1
 bfd fast-detect
```

インターフェイスでの OSPF3 への BFD の有効化

次に、OSPFv3 での BFD を特定のインターフェイスで設定する手順について説明します。この方法の手順は、コマンド モードが異なる点を除き、IS-IS および MPLS-TE での BFD を設定する手順と共通です。

（注） インターフェイス単位での BFD の設定は、OSPF、OSPFv3、および IS-IS のみでサポートされます。

```
Router# configure

/* Enter OSPF configuration mode to configure the OSPF routing process. */
Router(config)# router ospf3 0

/* Set the BFD minimum interval. The range is from 15 to 30000 milliseconds. */
Router(config-ospfv3)# bfd minimum-interval 6500

/* Set the BFD multiplier. */
Router(config-ospfv3)# bfd multiplier 7

/* Configure an Open Shortest Path First (OSPF) area. */
Router(config-ospfv3)# area 0

/* Enter interface configuration mode. */
Router(config-ospfv3-ar)# interface gigabitEthernet 0/5/0/1

/* Enable BFD to detect failures in the path between adjacent forwarding engines. */
```

```
Router(config-ospfv3-ar-if)# bfd fast-detect
Router(config-ospfv3-ar-if)# commit
Router(config-ospfv3-ar-if)# end
```

実行コンフィギュレーション

```
configure
  router ospf3 0
  bfd minimum-interval 6500
  bfd multiplier 7
   area 0
    interface gigabitEthernet 0/5/0/1
     bfd fast-detect
!
```

確認

適切なインターフェイスで BFD がイネーブルになっていることを確認します。

```
RP/0/RP0/CPU0:router(config-ospfv3-ar-if)#show run router ospf3

  router ospf3 0
  bfd minimum-interval 6500
  bfd multiplier 7
   area 0
    interface gigabitEthernet 0/5/0/1
     bfd fast-detect
```

BFD over BGP の有効化

BFD over BGP を設定するには、次の手順を実行します。次に、自律システム 65000 とネイバー 192.168.70.2 間での BFD を設定する例を示します。

```
Router# configure
Router(config)# router bgp 65000
Router(config-bgp)# bfd multiplier 2
Router(config-bgp)# bfd minimum-interval 20
Router(config-bgp)# neighbor 192.168.70.24
Router(config-bgp-nbr)# remote-as 2
Router(config-bgp-nbr)# bfd fast-detect
Router(config-bgp-nbr)# commit
Router(config-bgp-nbr)# end
```

実行コンフィギュレーション

```
router bgp 65000
 bfd multiplier 2
 bfd minimum-interval 20
 neighbor 192.168.70.24
 remote-as 2
 bfd fast-detect
 commit
 end
```

確認

BFD が BGP 上で有効になっていることを確認します。

```
Router# show run router bgp
router bgp 65000
  bfd multiplier 2
  bfd minimum-interval 20
  neighbor 192.168.70.24
  remote-as 2
  bfd fast-detect
```

IPv4 スタティック ルートでの BFD のイネーブル化

次に、IPv4 スタティック ルートでの BFD をイネーブルにする手順を示します。

実行コンフィギュレーション

確認

適切なインターフェイスで BFD がイネーブルになっていることを確認します。

IPv6 スタティック ルートでの BFD のイネーブル化

次に、IPv6 スタティック ルートでの BFD をイネーブルにする手順について説明します。

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

/* Enter static route configuration mode to configure static routing.  */
RP/0/RP0/CPU0:router(config)# router static

/* Enable BFD fast-detection on the specified IPv6 unicast destination address prefix
and on the forwarding next-hop address. */
/* BFD sessions are established with the next hop 2001:0DB8:D987:398:AE3:B39:333:783
when it becomes reachable. */

RP/0/RP0/CPU0:router(config-static)# address-family ipv6 unicast  2001:0DB8:C18:2:1::F/64
 2001:0DB8:D987:398:AE3:B39:333:783 bfd fast-detect minimum-interval 150 multiplier 4

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

実行コンフィギュレーション

```
configure
 router static
  address-family ipv6 unicast 2001:0DB8:C18:2:1::F/64 2001:0DB8:D987:398:AE3:B39:333:783
 bfd fast-detect minimum-interval 150 multiplier 4
  commit
```

確認

適切なインターフェイスで BFD がイネーブルになっていることを確認します。

```
RP/0/RP0/CPU0:router# show run router static address-family ipv6 unicast

 configure
 router static
```

```
address-family ipv6 unicast 2001:0DB8:C18:2:1::F/64 2001:0DB8:D987:398:AE3:B39:333:783
bfd fast-detect minimum-interval 150 multiplier 4
 commit
```

BFD カウンタのクリアと表示

次に、BFD パケット カウンタの表示およびクリアの手順について説明します。特定ノードまたは特定インターフェイスでホストされている BFD セッションのパケット カウンタをクリアすることができます。

```
RP/0/RP0/CPU0:router# show bfd counters all packet location 0/RP0/CPU0
RP/0/RP0/CPU0:router# clear bfd counters all packet location 0/RP0/CPU0
RP/0/RP0/CPU0:router# show bfd counters all packet location 0/RP0/CPU0
```

バンドル上の BFD

BFD over Bundle 機能により、BFD セッションは個々のバンドル メンバー リンクのステータスをモニタできます。BFD は、メンバー リンクの 1 つがダウンしたときにすぐにバンドル マネージャに通知し、バンドルが使用する帯域幅を減らします。

機能制限

BFD over Bundle 機能を使用する際の制限事項は次のとおりです。

- IETF モードでのみサポートされています。
- メイン バンドル インターフェイスでのみサポートされています。バンドル サブインターフェイスではサポートされていません。
- OSPF、ISIS、BGP などのルーティング プロトコルではサポートされていません。
- BFD タイマーが 3.3 ミリ秒に設定されている（最もアグレッシブなタイマー）場合、256 個のセッションを起動できます。
- BFD タイマーが 100 ミリ秒より長く設定されている場合、300 個の BFD セッションを同時に起動できます。
- BFD エコー モードと暗号化はサポートされていません。
- BFD ダンプニングはサポートされていません。

BFD over Bundle の設定

BFD over Bundle の設定には、次の手順が必要です。

- バンドルのモード、BFD パケット送信間隔、および障害検出時間の指定

（注） 宛先ルータで同じ設定手順を繰り返します。

```
/* Enable and Disable IPv6 checksum calculations for BFD on a router. */

Router(config-if)# bfd
Router(config-bfd-if)# ipv6 checksum disable
Router(config-bfd-if)# dampening disable
Router(config-bfd-if)# commit

/* Specify the mode, BFD packet transmission intervals, and failure detection times on
a bundle */

Router(config)# interface Bundle-Ether 3739
Router(config-if)# bfd mode ietf
Router(config-if)# bfd address-family ipv4 multiplier 3
Router(config-if)# bfd address-family ipv4 destination 10.23.1.2
Router(config-if)# bfd address-family ipv4 fast-detect
Router(config-if)# bfd address-family ipv4 minimum-interval 100
Router(config-if)# bfd address-family ipv6 multiplier 3
Router(config-if)# bfd address-family ipv6 destination 2001:DB8:1::2
Router(config-if)# bfd address-family ipv6 fast-detect
Router(config-if)# bfd address-family ipv6 minimum-interval 100
Router(config-if)# ipv4 address 10.23.1.1 255.255.255.252
Router(config-if)# ipv6 address 2001:DB8:1::2/120
Router(config-if)# load-interval 30
Router(config-if)# commit
Router(config)# interface TenGigE 0/0/0/0
Router(config-if)# bundle id 3739 mode active
```

実行コンフィギュレーション

```
bfd
 ipv6 checksum disable
 dampening disable!
 !

interface Bundle-Ether3739
 bfd mode ietf
 bfd address-family ipv4 multiplier 3
 bfd address-family ipv4 destination 10.23.1.2
 bfd address-family ipv4 fast-detect
 bfd address-family ipv4 minimum-interval 100
 bfd address-family ipv6 multiplier 3
 bfd address-family ipv6 destination 2001:DB8:1::2
 bfd address-family ipv6 fast-detect
 bfd address-family ipv6 minimum-interval 100
 ipv4 address 10.23.1.1 255.255.255.252
 ipv6 address 2001:DB8:1::2/120
 load-interval 30
 !

interface TenGigE 0/0/0/0
 bundle id 3739 mode active
```

確認

次の show コマンドの出力には、バンドル メンバの BFD セッションのステータスが表示されています。

```
/* Verify the details of the IPv4 BFD session in the source router. */

Router# show bfd session

Interface  Dest Addr  Local det time(int*mult)  State  Echo  Async   H/W   NPU
---------  ---------  --------  -------------   -----  ----  -----   ---   ---
Te0/0/0/0  10.23.1.2  0s(0s*0)  300ms(100ms*3)  UP     Yes               0/RP0/CPU0
BE3739     10.23.1.2  n/a       n/a             UP     No    n/a


/* Verify the details of the IPv4 BFD session in the destination router. */

Router# show bfd session

Interface   Dest Addr  Local det  time(int*mult) State   Echo   Async   H/W   NPU
----------  ---------  --------   -------------- -----  ----   -----   ---   ---
Te0/6/0/0   10.23.1.1  0s(0s*0)   300ms(100ms*3) UP      No     n/a
BE3739      10.23.1.1  n/a        n/a            UP      No     n/a


/* Verify the details of the IPv6 BFD session in the source router. */

Router# show bfd ipv6 session
Interface  Dest Addr  Local det  time(int*mult) State      H/W     NPU       Echo Async
---------- ---------- ---------- -------------- ------     ------  ------    ---- ----
Te0/0/0/0  10:23:1::2 Yes                       0/RP0/0s (0s*0) 00ms(100ms*3) UP

BE3739     10:23:1::2 No                        n/a        n/a      n/a       UP

/* Verify the details of the IPv6 BFD session in the destination router. */

Router# show bfd ipv6 session
Interface  Dest Addr  Local det time(int*mult) State    H/W NPU           Echo  Async
---------- ---------- --------- -------------- -------  --- ---           ----  ----
Te0/6/0/0  10:23:1::1 No        n/a            0s(0s*0)     300ms(100ms*3)  UP
BE3739     10:23:1::1 No        n/a            n/a          n/a             UP
```

BFD の透過性

Bidirectional Forwarding Detection（BFD）プロトコルは、設定されたタイマー値に応じて、1 秒未満でネットワークの障害を検出する単純な hello メカニズムです。

BFD セッションの両方のエンドポイントは、定期的に hello パケットを相互に送信します。これらのパケットを複数回受信しない場合は、セッションがダウンしていると見なされます。BFD は、あらゆるメディア タイプ、カプセル化、トポロジ、ルーティング プロトコル BGP、IS-IS、および OSPF の個別の高速 BFD ピア障害検出時間を提供します。

BFD の透過性機能を使用すると、L2VPN ネットワーク経由で接続されたカスタマーエッジデバイス間で BFD セッションを設定できます。これらの BFD セッションは、コアに対して透過

的です。たとえば、CE 間で交換される BFD パケットは、コア内のルータにドロップされることも、コア デバイス上でパントされることもありません。

この項では、イーサネット VPN（EVPN）仮想プライベート ワイヤ サービス（VPWS）で BFD の透過性を設定する方法を学習します。

イーサネット VPN 仮想プライベート ワイヤ サービス

EVPN VPWS（イーサネット VPN 仮想プライベート ワイヤ サービス）は、ポイントツーポイント サービス用の BGP コントロール プレーン ソリューションです。これにより、プロバイダー エッジ デバイスのペア間で EVPN インスタンスを確立するためのシグナリングおよびカプセル化技術が実装されます。

EVPN VPWS は、シングルホーミングとマルチホーミングの両方をサポートしています。

設定

次の項では、リモート LFA を使用して IP Fast Reroute を設定する手順について説明します。

- プロバイダー エッジ ルータでの L2VPN の設定
- カスタマー エッジ ルータでの BFD の設定

プロバイダー エッジ ルータでの L2VPN の設定

プロバイダー エッジ ルータでの L2VPN の設定

```
/* Enable IS-IS and configure routing level for an area. */
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# interface tengige 0/0/0/2.1
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)# 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)# bgp router-id 10.10.10.1
RP/0//CPU0:router(config-bgp)# address-family l2vpn evpn
RP/0//CPU0:router(config-bgp)# neighbor 192.0.2.1
RP/0//CPU0:router(config-bgp-nbr)# remote-as 100
RP/0//CPU0:router(config-bgp-nbr)# update-source Loopback 0
RP/0//CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
```

```
/* 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-nbr)# exit
RP/0//CPU0:router(config-bgp)# exit
RP/0//CPU0:router(config)# interface Bundle-Ether 199
RP/0//CPU0:router(config-if)# exit

/* Configure L2VPN Xconnect. */
RP/0//CPU0:router(config)# l2vpn
RP/0//CPU0:router(config-l2vpn)# router-id 10.10.10.1
RP/0//CPU0:router(config-l2vpn)# xconnect group bfdtr
RP/0//CPU0:router(config-l2vpn-xc)# p2p vpws-ce
RP/0//CPU0:router(config-l2vpn-xc-p2p)# interface TenGigE 0/0/0/1.1
RP/0//CPU0:ios(config-l2vpn-xc-p2p)# neighbor evpn evi 100 target 3 source 4
```

カスタマー エッジ ルータでの BFD の設定

カスタマー エッジ ルータでの BFD の設定

```
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)# address-family ipv4 unicast
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)# update-source TenGigE 0/0/0/16.1
RP/0//CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0//CPU0:router(config-bgp-nbr-af)#
```

実行コンフィギュレーション

このセクションでは、BFD 透過性設定を示します。

```
!
interface TenGigE 0/0/0/1.1
  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 l2vpn evpn
    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 0/0/0/1.1
    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
```

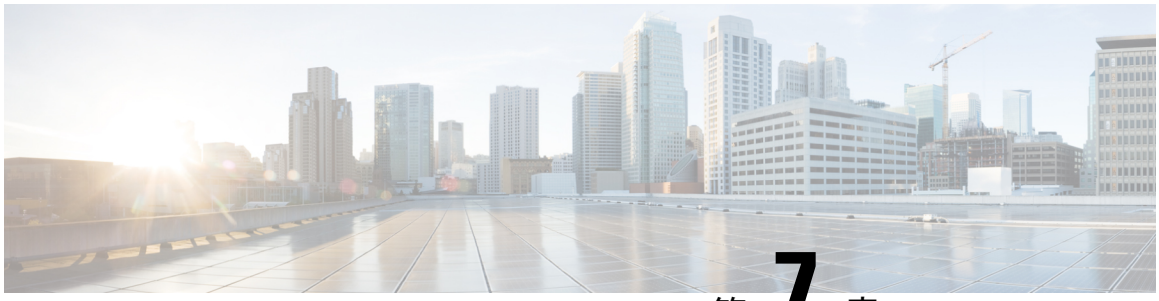
確認

次の項に示す show 出力には、BFD 透過性の設定の詳細とその設定のステータスが表示されます。

```
/* 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/RP0/CPU0
                                                      Yes   0/RP0/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/RP0/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:
                         Desired               Adjusted
  Client               Interval   Multiplier Interval   Multiplier
  -------------------- ---------------------- ---------------------
  bgp-default          10 ms      2          10 ms      2

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 location
0/RP0/CPU0

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
Iftype: 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/RP0/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
```

```
 Required echo rx interval: 0 ms, multiplier: 2, diag: None
 My discr: 2147483747, your discr: 2147483751, 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: 2147483751, your discr: 2147483747, state UP, D/F/P/C/A: 0/1/0/1/0

Tx Echo pkt :
 Version: 0, Local Discr: 2147483751, Sequence No: 0

History:
[Jan  4 03:00:19.272] Session (v1) state change, triggered by event 'Remote
    state init', from INIT to UP with current diag being None
[Jan  4 03:00:16.851] Session (v1) state change, triggered by event 'Remote
    state down', from DOWN to INIT with current diag being None
[Jan  4 03:00:16.509] Session (v1) state change, triggered by event 'Session
    create', from Unknown to DOWN with current diag being None
[Jan  4 03:00:16.509] Flag cleared: session creation is in-progress, currently
    set flags (0x80040)

Offload history:
[Jan  4 03:06:42.013] Packet punted to sw: Packet word0 : (0x20c80218),
desired_min_tx_interval 10000, required_min_rx_interval 10000, Last punted pkt
    required_min_rx_interval 10000
[Jan  4 03:06:42.003] Packet punted to sw: Packet word0 : (0x20d80218),
desired_min_tx_interval 10000, required_min_rx_interval 10000, Last punted pkt
    required_min_rx_interval 10000
[Jan  4 03:06:41.989] Packet punted to sw: Packet word0 : (0x20c80218),
desired_min_tx_interval 10000, required_min_rx_interval 10000, Last punted pkt
    required_min_rx_interval 10000
[Jan  4 03:06:41.980] Packet punted to sw: Packet word0 : (0x20d80218),
desired_min_tx_interval 10000, required_min_rx_interval 10000, Last punted pkt
    required_min_rx_interval 10000

Rx Counters and Timestamps :
Async valid packets received: count 5280
  [Jan  4 03:06:42.013]  [Jan  4 03:06:42.003]  [Jan  4 03:06:41.989]
Async valid packets while session is not in Up state: count 3
  [Jan  4 03:00:19.272]  [Jan  4 03:00:18.030]  [Jan  4 03:00:16.851]
```



第 7 章

Fast Reroute ループフリー代替の実装

Fast Reroute ループフリー代替機能により、障害のあるリンクを含むパケットを、リモート ループフリー代替（数ホップ離れている）までトンネリングすることができます。

- ループフリー代替による Fast Reroute の前提条件 （223 ページ）
- ループフリー代替による Fast Reroute の制約事項 （223 ページ）
- IS-IS および FRR （224 ページ）
- 修復パス （224 ページ）
- LFA の概要 （225 ページ）
- LFA の計算 （225 ページ）
- RIB とルーティング プロトコル間の連携 （226 ページ）
- リモート ループフリー代替による Fast Reroute （226 ページ）
- 設定 （228 ページ）

ループフリー代替による Fast Reroute の前提条件

- ループフリー代替による Fast Reroute 機能は、インターフェイスがポイントツーポイント インターフェイスである場合にのみ、インターフェイスを介して到達可能なパスを保護できます。
- LAN インターフェイスが 1 つのネイバーに物理的に接続されている場合、ループフリー代替（LFA）FRR 経由で保護するために、LAN インターフェイスをポイントツーポイント インターフェイスとして設定する必要があります。
- リモート ループフリー代替による Fast Reroute 機能の適切な展開のために、保護されたリンクも BFD で設定する必要があります。

ループフリー代替による Fast Reroute の制約事項

- ロード バランス サポートは、FRR で保護されたプレフィックスで利用可能ですが、50 ミリ秒のカットオーバーの時間は保証されません。

- 最大 8 個の FRR 保護のインターフェイスで同時にカットオーバーを実行することができます。
- LFA 計算は、同じレベルまたは領域に属するインターフェイスまたはリンクに制限されます。したがって、バックアップ LFA の計算時に同じ LAN 上のすべてのネイバーを除外すると、トポロジのサブセットで修復を使用できなくなる可能性があります。
- 物理インターフェイスと物理ポートチャネルインターフェイスおよびサブインターフェイスのみが保護されます。トンネルおよび仮想インターフェイスは保護されません。
- MPLS トラフィックのリモート LFA バックアップ パスは、LDP を使用してのみ設定できます。プレフィックス単位の保護のみがサポートされます。
- ボーダー ゲートウェイ プロトコル（BGP）プレフィックス独立コンバージェンス（PIC）と FRR は、同じプレフィックスに使用されない限り、同じインターフェイス上に設定できます。

IS-IS および FRR

ローカル リンクがネットワークで失敗した場合、IS-IS は、影響を受けるすべてのプレフィックスの新しいプライマリ ネクストホップルートを再計算します。これらのプレフィックスは、RIBおよび転送情報ベース（FIB）で更新されます。プライマリプレフィックスがフォワーディング プレーンで更新されるまで、影響を受けるプレフィックス宛てのトラフィックは廃棄されます。このプロセスには数百ミリ秒かかることがあります。

FRR で、IS-IS はプライマリ パスで障害が発生した場合に使用するために、フォワーディング プレーンに対する LFA ネクストホップ ルートを計算します。LFA はプレフィックスごとに計算されます。

特定のプライマリ パスに複数の LFA がある場合、IS-IS はプライマリ パスの単一 LFA を選ぶために、タイブレーク ルールを使用します。複数 LFA パスを持つプライマリ パスの場合、プレフィックスは LFA パス間で均等に分散されます。

修復パス

修復パスでは、ルーティングの遷移時にトラフィックが転送されます。リンクまたはルータに障害が発生すると、物理層の信号が失われるため、当初は隣接ルータしかこの障害を認識できません。ネットワーク内のその他すべてのルータは、この障害に関する情報がルーティング プロトコルによって伝播されるまで（これには数百ミリ秒かかる可能性があります）、この障害の性質と場所を認識しません。したがって、このネットワーク障害の影響を受けたパケットがそれぞれの宛先に到達するように準備する必要があります。

障害が発生したリンクに隣接するルータは、障害が発生したリンクを使用していた可能性のあるパケットに対して、一連の修復パスを使用します。これらの修復パスは、ルータが障害を検出してから、ルーティングの遷移が完了するまで使用されます。ルーティングの遷移が完了す

るまでに、ネットワーク内のすべてのルータは転送データを変更し、障害が発生したリンクはルーティングの計算から除外されます。

修復パスは、障害が検出されるとすぐにアクティブになるようにするために、障害を予測して事前計算されます。

LFA FRR 機能では次の修復パスを使用します。

- 等コストマルチパス（ECMP）は、宛先の等コストパス分割セットのメンバーとしてリンクを使用します。セットの他のメンバーは、リンクに障害が発生したときに代替パスを提供できます。
- LFA は、ループバックしないで宛先にパケットを送るネクストホップ ルートです。ダウンストリーム パスは LFA のサブセットです。

LFA の概要

LFA はプライマリ ネイバー以外のノードです。トラフィックは、ネットワーク障害発生後に LFA にリダイレクトされます。LFA は、失敗について認識せずに転送を決定します。

LFA は、トラフィックの転送に障害のある要素を使用したり、保護ノードを使用することはできません。LFA はループを発生させてはなりません。LFA は、インターフェイスがプライマリ パスとして使用できる限り、デフォルトでサポートされるすべてのインターフェイスでイネーブルになります。

プレフィックスごとの LFA を使用する利点は次のとおりです。

- プライマリ パスでリンクがダウンした場合、修復パスが移行中にトラフィックを転送します。
- プレフィックスごとの LFA を持つすべての宛先が保護されます。これにより、サブセット（障害の遠端のノード）のみが保護されない状態で残ります。

LFA の計算

プレフィックスごとに LFA を計算する汎用アルゴリズムについては、RFC 5286 を参照してください。IS-IS は、メモリ使用量を減らすための少量の変更とともに RFC 5286 を実装します。保護のプレフィックスを検証する前にすべてのネイバーの最短パス優先（SPF）計算を実行する代わりに、IS-IS は SPF 計算がネイバーごとに実行された後でプレフィックスを検査します。IS-IS は SPF 計算の実行後にプレフィックスを検査するため、IS-IS は SPF 計算がネイバーごとに実行された後も最適な修復パスを保持します。IS-IS では、すべてのネイバーに対する SPF の結果を保存する必要はありません。

RIB とルーティング プロトコル間の連携

ルーティング プロトコルは、タイブレーク アルゴリズムを実装して、プレフィックスの修復パスを計算します。計算の結果は、プライマリ パス付きの一連のプレフィックスになり、いくつかのプライマリ パスが修復パスに関連付けられます。

タイブレーク アルゴリズムは特定の条件を満たすか、または特定の属性を持つ LFA を考慮します。複数の LFA がある場合は、 **tie-break** キーワードを使用して **fast-reroute per-prefix** コマンドを設定します。ルールによってすべての候補LFAが除外される場合、そのルールはスキップされます。

プライマリ パスには、複数の LFA を設定できます。デフォルトのタイブレーク ルールを実装し、ユーザがこれらのルールを変更できるようにするには、ルーティングプロトコルが必要です。タイブレーク アルゴリズムの目的は、複数の候補 LFA を除外し、プレフィックス単位のプライマリ パスごとに 1 つの LFA を選択し、プライマリ パスが失敗したときに複数の候補 LFA でトラフィックを分散させることです。

タイブレーク ルールでは、すべての候補を除外することはできません。

タイブレークには、次の属性が使用されます。

- ダウンストリーム：保護された宛先へのメトリックが宛先へのノードを保護しているメトリックよりも低い候補を除外します。
- ラインカード分離：保護されたパスと同じラインカードを共有している候補を除外します。
- 共有リスク リンク グループ（SRLG）：保護されたパス SRLG のいずれかに属する候補を除外します。
- 負荷分散：保護されたパスを共有するプレフィックスで残りの候補を分散させます。
- 最低修復パス メトリック：保護されたプレフィックスへのメトリックが高い候補を除外します。
- ノードの保護：保護されたノードではない候補を除外します。
- プライマリ パス：ECMP ではない候補を除外します。
- セカンダリ パス：ECMP の候補を除外します。

リモート ループフリー代替による Fast Reroute

リモート ループフリー代替による Fast Reroute（FRR リモート LFA）機能により、障害のあるリンクを含むパケットを、リモートループフリー代替（数ホップ離れている）までトンネリングすることができます。

リンクやルータに障害が発生すると、分散ルーティングアルゴリズムによって障害を考慮した新しいルートが計算されます。計算のための時間をルーティングの遷移と呼びます。遷移が完

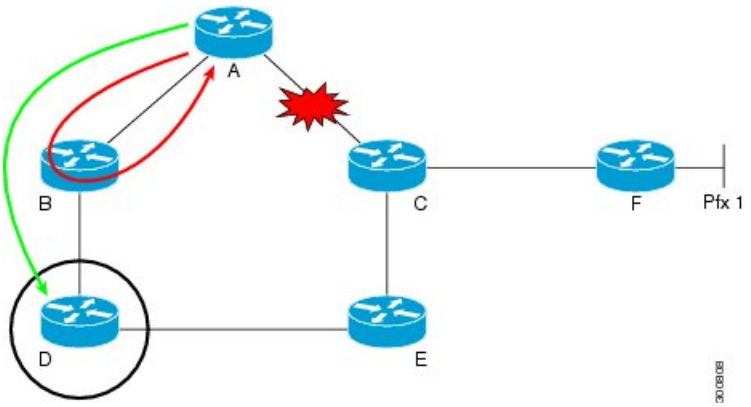
了し、すべてのルータがネットワーク上の共通のビューで収束されるまで、送信元と宛先のペア間の接続は中断されます。事前計算済みの代替ネクストホップを使用してルーティングの遷移時間を 50 ミリ秒より少なくするために、IP ループフリー代替（LFA）Fast Reroute（FRR）を使用できます。リンク障害の通知を受けると、ルータはトラフィック損失を減らすために、修復パスにすぐに切替えます。IGP/BGP コンバージェンスにおけるルーティング遷移には、最大数百ミリ秒かかる場合があることに注意してください。

IP ループフリー代替（LFA）Fast Reroute（FRR）は、修復パスの事前計算をサポートしています。Intermediate System-to-Intermediate System（IS-IS）ルーティング プロトコルによって、修復パスの計算が可能になります。結果の修復パスはルーティング情報ベース（RIB）に送信されます。Cisco Express Forwarding（旧 CEF）と Open Shortest Path First（OSPF）は、修復パスをインストールします。

IP ローカル LFA FRR では、IGP は直接接続されたネイバーのみを LFA バックアップ パスとして計算し、特定のプレフィックスのプライマリ パスを保護します。Label Distribution Protocol（LDP; ラベル配布プロトコル）は、保護されたプレフィックスのネクストホップを使用してラベル付けされたバックアップ LSP を設定します。一部のトポロジ（一般に使用されるリングベースのトポロジなど）は、LFA FRR で提供できない保護を必要とします。そのような場合は、LDP ベースの FRR リモート LFA 機能を使用します。この機能では、IGP が非直接接続ネイバー（数ホップ離れている）を LFA バックアップ パスとして計算し、特定のプレフィックスのプライマリ パスを保護します。LDP は、保護されたプレフィックスのリモート ネクストホップを使用してラベル付けされたバックアップ LSP を設定します。また、LDP は、リモート ノードから学習した LFA バックアップ ラベルを公開することなく、トラフィックをリモート ネクストホップにトンネリングする別のトランスポート LSP を設定します。

下の図 のトポロジについて考えます。

図 *6 :* リモート *LFA* による *FRR* とリング トポロジ



デバイス A は、F を宛先として B にネクストホップするトラフィックを送信しようと試みます。ノード C および F によってアドバタイズされているプレフィックスの LFA として、デバイス B を使用することはできません。ただし、ノード D は、保護ノード A に直接接続されていません。C によりアドバタイズされるプレフィックスを保護するには、障害中のリンクをトンネルが通過しない条件で、ノード A は、障害中の A - C リンクを含むパケットをノード D までトンネリングする必要があります。

FRR リモート LFA 機能により、障害のあるリンクを含むパケットを、リモート ループフリー代替（数ホップ離れている）までトンネリングすることができます。上の図で、AとDの間の緑色の矢印は、リモート LFA 機能によりルーピングをバイパスするために自動的に生成されたトンネルを表しています。

設定

LFA による FRR を設定するには、次のタスクを実行します。

ローカル LFA による FRR の設定

```
/* Configure FRR with local LFA using IS-IS */
Router# configure
Router(config)# router isis ring
Router(config)#  is-type level-1
Router(config-isis)# net 49.0001.0000.0000.0007.00
Router(config-isis)# nsf cisco
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# mpls traffic-eng level-1-2
Router(config-isis-af)# mpls traffic-eng router-id 10.7.7.7
Router(config-isis-af)# exit
Router(config-isis)# interface Loopback 0
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if)# exit
Router(config-isis)# interface TenGigabitEthernet 0/0/0/4
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-af)# fast-reroute per-prefix
Router(config-isis-af)# commit

/* Configure FRR with local LFA using OSPF*/
Router# configure
Router(config)# router ospf 50
Router(config-ospf)# router-id 10.1.1.1
Router(config-ospf)# address-family ipv4 unicast
Router(config-ospf-af)# area 0
Router(config-ospf-af)# mpls traffic-eng
Router(config-ospf-af)# interface Loopback 0
Router(config-ospf-af)# exit
Router(config-ospf)# interface HundredGigE0/9/0/0
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
```

リモート LFA によるリモート FRR を設定します。

```
/* Configure FRR with remote LFA using IS-IS */
Router# configure
Router(config)# router isis ring
Router(config)#  is-type level-1
Router(config-isis)# net 49.0001.0000.0000.0007.00
Router(config-isis)# nsf cisco
Router(config-isis)# address-family ipv4 unicast
```

```
Router(config-isis-af)# metric-style wide
Router(config-isis-af)# mpls traffic-eng level-1-2
Router(config-isis-af)# mpls traffic-eng router-id 10.7.7.7
Router(config-isis-af)# exit
Router(config-isis)# interface Loopback 0
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if)# exit
Router(config-isis)# interface TenGigabitEthernet 0/0/0/4
Router(config-isis-if)# point-to-point
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-af)# fast-reroute per-prefix remote-lfa
Router(config-isis-af)# fast-reroute per-prefix remote-lfa prefix-list /* The prefix-list
 option filters PQ node router ID based on prefix list */
Router(config-isis-af)# fast-reroute per-prefix remote-lfa prefix-list RLFA
Router(config-isis-af)# fast-reroute per-prefix remote-lfa tunnel mpls-ldp
Router(config-isis-af)# commit

/* Configure FRR with remote LFA using OSPF */
Router# configure
Router(config)# router ospf 50
Router(config-ospf)# router-id 10.1.1.1
Router(config-ospf)# address-family ipv4 unicast
Router(config-ospf-af)# area 0
Router(config-ospf-af)# mpls traffic-eng
Router(config-ospf-af)# interface Loopback 0
Router(config-ospf-af)# exit
Router(config-ospf)# interface HundredGigE0/9/0/0
Router(config-ospf-if)# fast-reroute per-prefix
Router(config-ospf-if)# fast-reroute per-prefix remote-lfa tunnel mpls-ldp
Router(config-ospf-if)# exit
Router(config-ospf)# exit
Router(config)# mpls traffic-eng router-id Loopback 0
```

実行コンフィギュレーション

この項では、ローカル LFA による FRR 構成を示します。

```
/* 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 HundredGigE0/9/0/0
  point-to-point
  address-family ipv4 unicast
fast-reroute per-prefix

/* FRR with local LFA with OSPF */
```

```
router ospf 50
 router-id 10.1.1.1
address-family ipv4 unicast
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface HundredGigE0/9/0/0
 fast-reroute per-prefix
  !
!
 mpls traffic-eng router-id loopback 0
!
```

この項では、リモート LFA による FRR 構成を示します。

```
/* FRR with remote  LFA with ISIS */
ipv4 prefix-list RLFA
 10 deny 3.3.3.3/32
 20 permit 0.0.0.0/0 le 32
router isis ring
 is-type level-1
 net 49.0001.0000.0000.0007.00
 nsf cisco
 address-family ipv4 unicast
 fast-reroute per-prefix remote-lfa prefix-list RLFA
  metric-style wide
  mpls traffic-eng level-1-2
  mpls traffic-eng router-id 10.7.7.7
 !
 interface Loopback0
  point-to-point
  address-family ipv4 unicast
  !
 !
 interface TenGigabitEthernet 0/0/0/4
  point-to-point
  address-family ipv4 unicast
fast-reroute per-prefix
fast-reroute per-prefix remote-lfa tunnel mpls-ldp


/* FRR with remote  LFA with OSPF */
router ospf 50
 router-id 10.1.1.1
address-family ipv4 unicast
 area 0
  mpls traffic-eng
  interface Loopback0
  !
  interface HundredGigE0/9/0/0
 fast-reroute per-prefix
fast-reroute per-prefix remote-lfa tunnel mpls-ldp
  !
!
 mpls traffic-eng router-id loopback 0
!
```

確認

次の項に示すshow出力には、リモートLFAによるFRR機能の設定の詳細とその設定のステータスが表示されます。

```
/* Verify the route summary information about the specified routing table. */

RP/0//CPU0:router# show route 10.3.3.3

Routing entry for 10.3.3.3/32
   Known via "isis 44", distance 115, metric 20, type level-1
   Installed Nov 15 19:43:13.367 for 00:00:34
   Routing Descriptor Blocks
     10.1.1.1, from 10.3.3.3, via TenGigE0/0/0/0, Backup (remote)
       Remote LFA is 10.9.9.9
      Route metric is 0
     10.1.1.2, from 10.3.3.3, via TenGigE0/7/0/3, Protected
       Route metric is 20
   No advertising protos.


/* Verify the MPLS LDP configuration. */

RP/0//CPU0:router# show running mpls ldp
Codes:
   - = GR label recovering, (!) = LFA FRR pure backup path
   {} = Label stack with multi-line output for a routing path
   G = GR, S = Stale, R = Remote LFA FRR backup

 Prefix           Label   Label(s)       Outgoing     Next Hop            Flags
                  In      Out            Interface                        G S R
 ---------------- ------ -------- -------- --------------- ----
 192.0.2.0/24     16019   { 16001        Te0/0/0/0    10.1.1.1        (!)     R
                            28006 }                  (10.9.9.9)
                          ImpNull        Te0/7/0/3    192.0.2.1
 192.0.2.1/32     16013   ImpNull        Te0/0/0/0    10.1.1.1
 192.0.1.0/32     16014   { 16001        Te0/0/0/0    10.1.1.1        (!)     R
                            16002 }                  (10.9.9.9)
                          ImpNull        Te0/7/0/3    192.0.2.2
 10.9.9.9/32      16012   16001          Te0/0/0/0    10.1.1.1
                          28006          Te0/7/0/3    192.0.2.1
 10.23.1.0/24     16018   16004          Te0/0/0/0    10.1.1.1        (!)
                          ImpNull        Te0/7/0/3    192.0.2.1
 10.34.1.0/24     16015   ImpNull        Te0/0/0/0    10.1.1.1
 10.0.0.1/32      16011  { 16001         Te0/0/0/0     10.1.1.1        (!)     R
                            16013 }                  (10.9.9.9)
                          16016          Te0/7/0/3    192.0.2.1
 10.100.0.2/32     16010   { 16001       Te0/0/0/0    10.1.1.1         (!)     R


/* Verify whether RLFA filtering is active */
RP/0/0/CPU0:Router #show isis fast-reroute 1.0.0.2/32 detail

L2 1.0.0.2/32 [20/115] medium priority
     via 1.2.0.2, GigabitEthernet0/0/0/0, R2, Weight: 0
       Backup path: R-LFA, via R3 [1.0.0.3], via 1.4.1.2, GigabitEthernet0/0/0/1 R4,
Weight: 0, Metric: 20 /*3.3.3.3 is filtered out, and another address is picked when RLFA
 filtering is active */
       P: No, TM: 20, LC: No, NP: No, D: No, SRLG: Yes
     src R2.00-00, 1.0.0.2
```