



Cisco ASR 9000 シリーズ ルータ への ルーティン グ ポリシー の 実装

ルーティング ポリシーは、ルート の 検 査、ル ー ト の フィルタ、ま た 場 合 に よ っ て は、ル ー ト の 属 性 の 変 更 を ル ー タ に 指 示 し て、ル ー ト が ピ ア か ら 受 け 入 れ ら れ る よ う に、ピ ア に ア ド バ タ イ ズ さ れ る よ う に、ま た は 1 つ の ル ー テ ィ ン グ プ ロ ト コ ル か ら 別 の ル ー テ ィ ン グ プ ロ ト コ ル に 再 配 信 さ れ る よ う に 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 さ れ た ル ー テ ィ ン グ ポ リ シ ー に 基 づ い て、ア ド バ タ イ ズ、集 約、廃 棄、配 信、エ ク ス ポ ー ト、保 持、イ ン ポ ー ト、再 配 信 お よ び 変 更 の 実 行 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) は、す べ て の ル ー テ ィ ン グ ポ リ シ ー の ニ ー ズ が 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。従 来 の ル ー テ ィ ン グ ポ リ シ ー 設 定 メ ソ ッ ド に 固 有 の 冗 長 性 を RPL で は 大 幅 に 減 ら し ま す。RPL は、ル ー テ ィ ン グ ポ リ シ ー 設 定 を 簡 素 化 し、こ れ ら の 設 定 を 保 存 や 処 理 す る た め に 必 要 な シ ス テ ム リ ソ ー ス を 削 減 し て、ト ラ ブ ル シ ュ ー テ ィ ン グ を 容 易 に し ま す。



(注)

Cisco IOS XR ソフトウェアにおけるルーティング ポリシーの詳細、およびこのモジュールに記載されているルーティング ポリシー コマンドの詳細については、このモジュールの[関連資料](#)、(80 ページ) の項を参照してください。設定タスクを実行中に表示される他のコマンドのマニュアルを見つけるには、オンラインで『Cisco ASR 9000 Series Aggregation Services Router Commands Master List』を検索してください。

ル ー テ ィ ン グ ポ リ シ ー の 実 装 の 機 能 履 歴

リリース	変更箇所
リリース 3.7.2	この機能が導入されました。
リリース 3.9.0	すべての接続点で、パラメータ化がサポートされました。

リリース	変更箇所
リリース 4.2.0	次の機能が追加されました。 <ul style="list-style-type: none"> • 階層的な条件 • 条件ポリシーの適用
リリース 4.2.1	次の機能が導入されました。 <ul style="list-style-type: none"> • 拡張プレフィックス長操作。 • ネストされたワイルドカード適用ポリシー。 • XML を使用したルーティング ポリシー言語セット要素の編集。 • bgp export および bgp import の接続点の「med」属性の有効な演算子として「set」をサポートします。

- [ルーティング ポリシー実装の前提条件, 2 ページ](#)
- [ルーティング ポリシー実装の制約事項, 3 ページ](#)
- [ルーティング ポリシーの実装に関する情報, 3 ページ](#)
- [ルーティング ポリシーの実装方法, 72 ページ](#)
- [ルーティング ポリシーの実装の設定例, 77 ページ](#)
- [その他の参考資料, 80 ページ](#)

ルーティング ポリシー実装の前提条件

Cisco IOS XR ソフトウェアへのルーティング ポリシーの実装の前提条件は、次のとおりです。

- 適切なタスク ID を含むタスク グループに関連付けられているユーザ グループに属している必要があります。このコマンドリファレンスには、各コマンドに必要なタスク ID が含まれます。ユーザグループの割り当てが原因でコマンドを使用できないと考えられる場合、AAA 管理者に連絡してください。
- ボーダー ゲートウェイ プロトコル (BGP) 、Integrated Intermediate System-to-Intermediate System (IS-IS) 、または Open Shortest Path First (OSPF) がネットワークに設定されている必要があります。

ルーティング ポリシー実装の制約事項

Cisco IOS XR ソフトウェアへのルーティング ポリシー言語の実装を取り扱う際には、次の制約事項が適用されます。

- 個別のポリシー定義では、最大 1000 のステートメントがサポートされています。1 つのポリシー内の合計ステートメント数は、階層型ポリシー構造を使用して 4000 のステートメントまで拡張できます。ただし、この制約事項の適用は **apply** ステートメントの使用に制限されています。
- 接続点に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とき、**commit** 操作は次の場合には実行できません。
 - 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。

ルーティング ポリシーの実装に関する情報

RPL を実装するには、次の概念を理解しておく必要があります。

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、維持する必要が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、`ifthen`、`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、セットという用語は、数学的意味において固有の要素の非順序の集合の意味で使用されます。ポリシー言語は、一致目的の値のグループのコンテナとしてセットを提供します。セットは条件式で使用します。セットの要素はカンマで区切ります。ヌル（空）のセットも使用できます。

次の例では、

```
prefix-set backup-routes
# currently no backup routes are defined
end-set
```

次のような条件は、

```
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。

セットには、[as-path-set](#) (6 ページ)、[community-set](#) (7 ページ)、[extcommunity-set](#) (7 ページ)、[prefix-set](#) (11 ページ)、および [rd-set](#) (13 ページ) の 5 種類があります。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ります。これらの比較ができるように、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に相当しますが、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、次の各項で説明するセットタイプについてを参照してください。通常、インラインセットの構文は、次のような括弧で囲まれたカンマ区切りのリストです ([element-entry](#), [element-entry](#), [element-entry](#), ...[element-entry](#))。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、次のとおりです。

- **dfa-regex** : DFA スタイル正規表現
- **ios-regex** : 従来の IOS スタイル正規表現
- **length** : BGP AS パスの長さ
- **neighbor-is** : BGP AS パス ネイバー
- **originates-from** : BGP AS パスの発信元
- **passes-through** : BGP AS パスのパススルー
- **unique-length** : 重複を無視する BGP AS パスの長さ

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せん。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。詳細については、[パラメータ化](#)、(17 ページ) を参照してください。

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 cost の名前付き形式

cost セットは、コスト EIGRP コスト コミュニティ タイプの拡張コミュニティ 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 cost a_cost_set
  IGP:1:10
end-set
```

次のオプションが拡張コミュニティセット Cost でサポートされています。

```
RP/0/RSP0/CPU0:router(config)#extcommunity-set cost cost_set
RP/0/RSP0/CPU0:router(config-ext)#?
#-remark      Remark beginning with '#'
<0-255>       decimal number
abort         Discard RPL definition and return to top level config
end-set       End of set definition
exit         Exit from this submode
igp:         Cost Community with IGP as point of insertion
pre-bestpath: Cost Community with Pre-Bestpath as point of insertion
show         Show partial RPL configuration
```

オプション	説明
#-remark	「#」ではじまる注記
<0-255>	10 進数
abort	RPL 定義を廃棄して、top level config に戻る
end-set	セット定義の終了
exit	このサブモードを終了
igp:	IGP を挿入ポイントとして使用するコスト コミュニティ
pre-bestpath:	Pre-Bestpath を挿入ポイントとして使用するコスト コミュニティ
show	一部の RPL 設定を表示

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 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)
($ipaddr:666, 1234:$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/RSP0/CPU0:router(config)#extcommunity-set rt rt_set
RP/0/RSP0/CPU0:router(config-ext)#?
#-remark      Remark beginning with '#'
*             Wildcard (any community or part thereof)
<1-4294967295> 32-bit decimal number
<1-65535>     16-bit decimal number
A.B.C.D/M:N   Extended community - IPv4 prefix format
A.B.C.D:N     Extended community - IPv4 format
ASN:N        Extended community - ASPLAIN format
X.Y:N        Extended community - ASDOT format
abort        Discard RPL definition and return to top level config
dfa-regex    DFA style regular expression
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

オプション	説明
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

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
  1.1.1.1:100,
  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/RSP0/CPU0:router(config)#extcommunity-set soo soo_set
RP/0/RSP0/CPU0:router(config-ext)#?
#-remark      Remark beginning with '#'
*             Wildcard (any community or part thereof)
<1-4294967295> 32-bit decimal number
<1-65535>     16-bit decimal number
A.B.C.D/M:N   Extended community - IPv4 prefix format
A.B.C.D:N     Extended community - IPv4 format
ASN:N         Extended community - ASPLAIN format
X.Y:N         Extended community - ASDOT format
abort         Discard RPL definition and return to top level config
dfa-regex     DFA style regular expression
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（**greater than or equal to**）のニーモニック）で表され、その後には 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（**less than or equal to**）のニーモニック）で表され、その後には 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（**equal 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 番目の要素は、唯一の有効値 10.0.2.0/24 と一致します。3 番目の要素は、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 番目の要素は、10.0.4.0/24 ~ 10.0.4.240/28 の範囲の値と一致します。5 番目の要素は、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番目の要素は、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。

拡張プレフィックス長操作

`prefix-set` における拡張プレフィックス長操作のサポートは、プレフィックス一致指定での `ge` セマンティックを使用する `prefix-range` を拡張します。これは、プレフィックス 0.0.0.0/0, 0.0.0.0/1, 0.0.0.0/2, ..., 0.0.0.0/32 と一致する単一のエントリを保持する要求に応えます。`prefix-length` は、`ge` セマンティックを使用して操作できます。たとえば、`prefix-set (0.0.0.0/30 ge 0 le 32)` は、0.0.0.0/0 ~ 0.0.0.3/32 の範囲内のすべてのプレフィックスと一致します。これにより、単一の `prefix-set` エントリ 0.0.0.0/32 ge 0 le 32 は、プレフィックス 0.0.0.0/0, 0.0.0.0/1, 0.0.0.0/2, ..., 0.0.0.0/32 と一致します。

次に、IPv4 プレフィックス構文と対応するマスク長範囲を含むプレフィックス範囲を示します。

- <A.B.C.D>/<len> ge <G> le <L>
 - <A.B.C.D>/[<len>..<G>] (if <len> is lesser than <G>)
 - <A.B.C.D>/[<G>..<len>] (if <len> is greater than <G>)
- <A.B.C.D>/<len> ge <G>
 - <A.B.C.D>/[<len>..<G>] (if <len> is lesser than <G>)
 - <A.B.C.D>/[<G>..<len>] (if <len> is greater than <G>)
- <A.B.C.D>/<len> eq <E>
 - <A.B.C.D>/[<len>..<E>] (if <len> is lesser than <E>)
 - <A.B.C.D>/[<E>..<len>] (if <len> is greater than <E>)

rd-set

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 ポリシー言語には、定義、変更、およびポリシーの使用に利用する、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自体の4つの主要なコンポーネントが含まれています。

設定フロントエンド (CLI) は、ポリシーを定義および変更するメカニズムです。この設定は、その後通常の保存方法を使用してルータに保存され、通常設定の **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つめのコンポーネントであるポリシーリポジトリには、いくつかの役割があります。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番めに、ポリシーの検証の大半を実行して、定義されたポリシーが実際に正常に実行できることを確認します。3番めに、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つめ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せん。

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、適切なときに実行エンジンを呼び出して指定のポリシーを指定のルートに適用してから、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法の例について説明します。ルーティング ポリシー言語の実装方法の詳細については、[ルーティング ポリシーの実装方法](#)、(72 ページ) を参照してください。

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 番号のルートの無視

次に、3 つの部分に分けてポリシー定義の例を示します。最初に、**as-path-set** コマンドは、AS パスの一致対象にする 3 つの正規表現を定義します。2 番めに、**route-policy** コマンドは、AS パスセットをルートに適用します。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 から受信 (インポート) したルート上の `ignore_path_as` という名前のポリシーを検索します。

```
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 に基づくコミュニティの設定

次に、ポリシーがルートの MED をテストして、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、ポリシーはコミュニティ 123:456 をルートに追加します。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 という名前の拡張コミュニティ 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
```

ルーティング ポリシーの設定の基本

ルート ポリシーは、**route-policy** キーワードおよび **end-policy** キーワードとともに括弧書きされる一連のステートメントと式によって構成されます。個別のコマンド（1行に1つのコマンド）の集合ではなく、ルート ポリシー内のステートメントには相互に関連するコンテキストがあります。そのため、個別のコマンドを各行に記すのではなく、各ポリシーまたはセットは独立した設定オブジェクトとして、1つのユニットとして使用、入力、操作できます。

ポリシー設定の各行は論理サブユニットです。 **then**、**else**、**end-policy** キーワードの後ろには、少なくとも1つの新しい行を続ける必要があります。また、パラメータリストの閉じ括弧、および 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またはプレフィックスセットへの参照にある名前文字列の後ろには新しい行が必要です。ルート ポリシー、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、またはプレフィックスセットの定義の前には、少なくとも新しい行が1行必要です。アクションステートメントの後ろには1行以上の新しい行を続けることができます。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、プレフィックスセットのカンマ区切りの後ろには1行以上の新しい行を続けることができます。新しい行はポリシー式の論理ユニットの最後に記される必要があります。他の場所に記すことはできません。

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。ポリシー階層は常に維持されるため、`param-example` の定義が変更されると、`origin_10` と `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

パラメータ化、(17 ページ) で説明したように `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 `glbpathtype` 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と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つの値だけが含まれる場合は、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、複数の値が含ま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番めの例として、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時、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。ポリシーを定義するときに参照するすべてのポリシーが、設定に存在してい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、未定義ポリシーバーへの参照があるポリシー例を着信 BGP ポリシーに付加しようとする、ポリシーバーが存在しないため、設定の試行は拒否されます。

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。

ポリシーバーは存在するが、その中にステートメント、アクション、または処理が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。そのため、ポリシーブロックのデフォルトアクションであるドロップに従います。

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、設定の変更は、該当する設定を完全に削除してから再度設定を入力することによって実行されてきました。しかし、これによりポリシーが付加されていないためにデフォルトアクションが実行される期間ができます。RPL は、アトミック変更のメカニズムを備えているため、ポリシーが再宣言された、またはテキストエディタを使用して編集された場合に、新しい設定がすぐに適用されるようになります。これにより、使用中のポリシーは指定の接続点において適用されているポリシーがない期間が生じることなく変更できるようになります。

属性比較とアクションの検証

ポリシーリポジトリは、各接続点で有効な属性、アクションおよび比較を認識しています。ポリシーが付加されたとき、これらのアクションおよび比較はその特定の接続点の機能に対して検証されます。例として、次のポリシー定義があります。

```
route-policy bad
set med 100
```

```
set level level-1-2
set ospf-metric 200
end-policy
```

このポリシーは、BGP 属性 **med**、IS-IS 属性レベル、および OSPF 属性コストを設定するアクションの実行を試行します。システムは、このようなポリシーの定義を許可しますが、このようなポリシーの付加は許可しません。不正なポリシーを定義した後で、BGP 設定ステートメント **neighbor 1.2.3.4 address-family ipv4 unicast route-policy bad in** を使用して、ポリシーを着信 BGP ポリシーとして付加しようとする、システムはこの設定の試行を拒否します。この拒否は、ポリシーをチェックする検証プロセスの結果であり、BGP は MED を設定できるが、レベルおよびコストがそれぞれ IS-IS および OSPF 属性であるため、レベルまたはコストを設定する方法がありません。実行できないアクションを黙示的に省く代わりに、システムはユーザに対してエラーを生成します。同様に、存在しない属性を変更、または存在しない属性との比較の試行を発生させる場合のように、接続点で使用中の有効なポリシーは変更できません。ベリファイアは、存在しない属性があるかをテストして、このような設定の試行を拒否します。

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

注記とは、ポリシー設定に付加されたテキストのことですが、ポリシー言語パーサーでは無視されます。注記は、ポリシーの一部を記述することに役立ちます。注記の構文は、ポンド記号 (#) が各行の先頭に付いたテキストです。

```
# 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 の注記はリポートしても持続し、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する場合、デフォルトで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続けます。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
  endif
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
```

默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の機能と組み合わせることによって、**apply** コマンドは反復設定を削減するための優れたツールになります。

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を検証されます。

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进行处理する接続点では、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。

属性および動作の詳細については、『Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference』を参照してください。

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。

Aggregation

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をもとに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。

Dampening

dampening 接続点は、BGP 内のデフォルトのルート ダンプニング挙動を制御します。関連ピア上のより具体的なポリシーに上書きされないかぎり、BGP 内のルートはすべて関連ポリシーを適用してダンプニング属性を設定します。

次のポリシーは、BGP IPv4 ユニキャストルートにダンプニング値を設定します。/25 より具体的なこれらのルートでは、ダンプされた後の回復に、/25 ほど具体的でないルートよりも長くかかります。

```
route-policy sample_damp
  if destination in (0.0.0.0/0 ge 25) then
    set dampening half-life 30 others default
  else
```

```

        set dampening halflife 20 others default
    endif
end-policy

router bgp 2
  address-family ipv4 unicast
    bgp dampening route-policy sample_damp
  .
  .
  .

```

Default Originate

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、ピアまたはピアのグループに送る BGP ルートを選択します。ルートは、関連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で選択されます。ポリシーをパスするルートすべてが、アップデートとしてピアまたはピアのグループに送られます。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。範囲 2:100 ~ 2:200 にあるコミュニティタグが付されたルートは、MED が 100、コミュニティが 2:666 で送信されます。残りのルートは MED が 200、コミュニティが 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。

次の例に、/24 よりさらに具体的なルートに対して **well-known** コミュニティ **no-export** を設定する、ネットワーク接続点でアタッチされるルート ポリシーの例を示します。

```

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せん。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。その後、このコマンドは変更前と変更後の各ルートの状態を、次の例に示すように表示します。

```

RP/0/RSP0
/CPU0:router# 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
!
RP/0/RSP0
/CPU0:router# 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
  ....

RP/0/RSP0
/CPU0:router# 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。このようにして、ルータオペレータはマッチング操作を使用して異なる BGP ルート属性のセットを選択し、トラッキングの対象となるそれぞれ異なるクラスのルートに対して別のトラフィック索引を設定できます。

次の例に、自律システム 10.33 を始点とする IPv4 ユニキャストルートにトラフィック索引として 10 を設定する方法を示します。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

```
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 ターゲット値 10:91 のルートの場合、ネクストホップは 172.16.0.1 に設定されます。ルート ターゲット値 11:92 のルートの場合、ネクストホップは 172.16.0.2 に設定されます。ルートが Site-of-Origin (SoO) 値 10:111111 または 10:111222 を持つ場合、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
  .
  .
  .
```



(注) bgp import 接続点では、「Set」は「med」属性の有効な演算子です。

Export

export 接続点を使用して、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を制御できます。

レイヤ 3 VPN ネットワークの場合、エクスポート ルート ターゲットは VRF IPv4 ルートが VPN IPv4 ルートに返還されるときに VPN IPv4 ルートに追加され、他の PE ルータ（または VRF から PE ルータ内の別の VRF へのフロー）に MP-iBGP を通じてアドバタイズされます。

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
  .
  .
  .
```



(注) bgp export 接続点では、「Set」は「med」属性の有効な演算子です。

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ます、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 が保持するように設定することで、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 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
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。

次の例に、プレフィックス 0.0.0.0 をプレフィックス長 0 でパスするルート ポリシーの設定方法を示します。ラベル割り当てが発生するのはプレフィックス 0.0.0.0 が存在する場合だけです。

```
route-policy label_policy
  if destination in (0.0.0.0/0) then
    pass
  endif
end-policy

router bgp 2
  vrf vrf1
    rd auto
  address-family ipv4 unicast
    allocate-label route-policy label-policy
  .
  .
  .
```

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に、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。このインバウンドフィルタリングに加え、BGP はこれらのプレフィックス エントリを許可または拒否とともに上流ネイバーに送り、ネイバーはアップデートを宛先に送る前にフィルタできます。

```
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、プレフィックス ベースのマッチング操作をより詳細に制御できます。この接続点は、通常、ルートのプレフィックスに基づいたさまざまな BGP コマンドのデバッグ出力のフィルタリングに使用します。

次の例に、プレフィックス長 8 のプレフィックス **20.0.0.0** だけをパスするルート ポリシーの設定方法を示します。こうして、デバッグ出力にはそのプレフィックスのものだけが表示されます。

```
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 1: BGP 属性と演算子

接続点	属性	一致	設定
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	設定
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	設定
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 (ダンプニングを制御する値の設定については、 Dampening , (31 ページ) を参照してください)
	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	設定
export (VRF)	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 rt	is-empty matches-any matches-every	set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every	—
	local-preference	is、 ge、 le、 eq	set
	med	is、 eg、 ge、 le	set
	next-hop	in	—
	origin	is	—
	source	in	—
weight	—	set	

接続点	属性	一致	設定
import (VRF)	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	—
	extcommunity soo	is-empty matches-any matches-every	—
	local-preference	is、 ge、 le、 eq	—
	med	is、 eg、 ge、 le	set
	next-hop	in	set set peer address set destination vrf
	origin	is	—
	source	in	—

接続点	属性	一致	設定
neighbor-in	as-path	in is-local length neighbor-is originates-from passes-through unique-length	prepend prepend most-recent replace
	as-path-length	is、 ge、 le、 eq	—
	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

接続点	属性	一致	設定
	extcommunity rt	is-empty matches-any matches-every	set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every	—
	local-preference	is、 ge、 le、 eq	set
	med	is、 eg、 ge、 le	set set + set -
	next-hop	in	set set peer address
	origin	is	set
	source	in	—
	weight	—	set

接続点	属性	一致	設定
neighbor-out	as-path	in is-local length neighbor-is originates-from passes-through unique-length	prepend prepend most-recent replace
	as-path-length	is、 ge、 le、 eq	—
	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	set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every	—
local-preference	is、 ge、 le、 eq	set	

接続点	属性	一致	設定
	med	is、eg、ge、le	set set + set - set max-unreachable set igp-cost
	next-hop	in	set set self
	origin	is	set
	path-type	is	—
	rd	in	—
	source	in	—
	unsuppress-route	—	unsuppress-route
	vpn-distinguisher	—	set
neighbor-orf	orf-prefix	in	—

接続点	属性	一致	設定
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	設定
redistribute	as-path	—	prepend
	community	—	set set additive delete in delete not in delete all
	destination	in	—
	extcommunity cost	—	set set 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	—

接続点	属性	一致	設定
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	—
	extcommunity soo	is-empty matches-any matches-every	—
	med	is、 eg、 ge、 le	—
	next-hop	in	—
	origin	is	—
source	in	—	

接続点	属性	一致	設定
table-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	—
	community	is-empty matches-any matches-every	—
	local-preference	is、 ge、 le、 eq	—
	destination	in	—
	med	is、 eg、 ge、 le	—
	next-hop	in	—
	origin	is	—
	rib-metric	—	set
	source	in	—
	tag	—	set
traffic-index	—	set	

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、元の値を提供するよう設定された **igp-cost** が存在する場合に意味を持ちます。

この表では、有効な操作と有効になる環境をまとめます。

表 2: 接続点により制限される BGP 操作

コマンド	import	export	aggregation	redistribution
prepend as-path most-recent	eBGP のみ	eBGP のみ	n/a	n/a
replace as-path	eBGP のみ	eBGP のみ	n/a	n/a
set med igp-cost	禁止	eBGP のみ	禁止	禁止
set weight	n/a	禁止	n/a	n/a
suppress	禁止	禁止	n/a	禁止

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。

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
  .
  .
```

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 インスタンス 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 接続点を使用して、インバウンド OSPF タイプ 3 サマリー LSA (Link-State Advertisement) 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-out 接続点を使用して、アウトバウンド 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、アナウンスされます。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、ドロップされます。

```

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 3: **OSPF** 属性と演算子

接続点	属性	一致	設定
default-information originate	ospf-metric	—	set
	metric-type	—	set
	tag	—	set
	rib-has-route	in	—
redistribute	destination	in	—
	metric-type	—	set
	ospf-metric	—	set
	next-hop	in	—
	mpls-label	route-has-label	—
	route-type	is	—
	tag	is、eq、ge、le	set

接続点	属性	一致	設定
area-in	destination	in	—
area-out	destination	in	—

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。

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` を OSPFv3 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、`2001::/96` 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 ospfv3-originate
  if rib-has-route in (2001::/96) then
    pass
  endif
end-policy

router ospfv3 1
  default-information originate policy ospfv3-originate
  :
```

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 4: *OSPFv3* 属性と演算子

接続点	属性	一致	設定
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。

Redistribute

IS-IS 内の `redistribute` 接続点を使用して、他のプロトコルからのルートを IS-IS によって再アドバタイズできます。ポリシーは IS-IS に再配布するルートの種類を選択するための制御構造のセットです。ポリシーでは、ルートを挿入する IS-IS レベルおよびそのときのメトリック値も制御可能です。

次の例に、ポリシー `ISIS-redist` を使用してルートを IS-IS インスタンス 1 から IS-IS インスタンス `instance_10` へ再配布する方法を示します。このポリシーは、再配布されるルートすべてのレベルを `level-1-2` に設定します。タグ 10 を持つ OSPF ルートはメトリックが 100 に設定され、タグ 20 を持つ IS-IS ルートは IS-IS メトリックが 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 IS-IS データベースに再配布されません。

```
route-policy ISIS-redist
  set level level-1-2
  if tag eq 10 then
    set isis-metric 100
  elseif tag eq 20 then
    set isis-metric 200
  else
    drop
  endif
end-policy

router isis instance_10
  address-family ipv4 unicast
    redistribute ospf 1 policy ISIS-redist
  .
  .
  .
```

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
  .
```

IS-IS 属性と演算子

この表では、接続点ごとの IS-IS 属性と演算子をまとめます。

表 5: IS-IS 属性と演算子

接続点	属性	一致	設定
redistribution	tag	is、eq、ge、le	set
	route-type	is	—
	destination	in	—
	next-hop	in	—
	mpls-label	route-has-label	—
	level	—	set
	isis-metric	—	set
	metric-type	—	set
default-information originate	rib-has-route	in	—
	level	—	set
	isis-metric	—	set
	tag	—	set
inter-area-propagate	destination	in	—

EIGRP ポリシー接続点

この項では、それぞれの EIGRP ポリシー接続点について説明し、EIGRP 属性と演算子の概要を示します。

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
  .
  .
  .
```

Default-Accept-Out

default-accept-out 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、10.10.0.0/16 に一致するすべてのルート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-out
  if destination in (10
.10.0.0/16) then
    pass
  endif
end-policy
!
router eigrp 100
  address-family ipv4
    default-information allowed out route-policy eigrp-cd-policy-out
  .
  .
  .
```

Policy-In

policy-in 接続点を使用して、インバウンド EIGRP ルートのフィルタリングと修正が行えます。このポリシーは、インターフェイスインバウンドルートポリシーを持たないすべてのインターフェイスに適用されます。

次の例に、EIGRP でのコマンドを示します。

```
router eigrp 100
  address-family ipv4
    route-policy global-policy-in in
  .
  .
  .
```

Policy-Out

policy-out 接続点を使用して、アウトバウンドEIGRPルートのフィルタリングと修正が行えます。このポリシーは、インターフェイスアウトバウンドルートポリシーを持たないすべてのインターフェイスに適用されます。

次の例に、EIGRP でのコマンドを示します。

```
router eigrp 100
  address-family ipv4
    route-policy global-policy-out out
  .
  .
  .
```

If-Policy-In

if-policy-in 接続点を使用して、特定の EIGRP インターフェイス上で受け取るルートのフィルタリングが行えます。次の例に、Packet-over-SONET/SDH (POS) インターフェイス 0/2/0/3 のインバウンドポリシーを示します。

```
router eigrp 100
  address-family ipv4
    interface POS0/2/0/3
      route-policy if-filter-policy-in in
  .
  .
  .
```

If-Policy-Out

if-policy-out 接続点を使用して、特定の EIGRP インターフェイス上で送出するルートのフィルタリングが行えます。次の例に、Packet-over-SONET/SDH (POS) インターフェイス 0/2/0/3 のアウトバウンドポリシーを示します。

```
router eigrp 100
  address-family ipv4
    interface POS0/2/0/3
      route-policy if-filter-policy-out out
  .
  .
  .
```

Redistribute

EIGRP 内の **redistribute** 接続点によって、他のルーティングプロトコルから再配布されたルートのフィルタリングや、ルートを EIGRP データベースにインストールする前に一部のルーティングパラメータを変更することが可能です。次の例に、RIP ルートの EIGRP への再配布をフィルタするポリシーを示します。

```
router-policy redistribute-rip
  if destination in (100.1.1.0/24) then
    set eigrp-metric 5000000 4000 150 30 2000
```

```

    else
      set tag 200
    endif
  end-policy

router eigrp 100
  address-family ipv4
    redistribute rip route-policy redistribute-rip
  .
  .
  .

```

EIGRP 属性と演算子

この表では、接続点ごとの EIGRP 属性と演算子をまとめます。

表 6 : EIGRP 属性と演算子

接続点	属性	一致	設定
default-accept-in	destination	in	—
default-accept-out	destination	in	—
if-policy-in	destination	in	—
	next-hop	in	—
	eigrp-metric	—	add、 set
	tag	is、 eq、 ge、 le	set
if-policy-out	destination	in	—
	next-hop	in	—
	protocol	is、 in	—
	eigrp-metric	—	add、 set
	tag	is、 eq、 ge、 le	set
policy-in	destination	in	—
	next-hop	in	—
	eigrp-metric	—	add、 set
	tag	is、 eq、 ge、 le	set

接続点	属性	一致	設定
policy-out	destination	in	—
	next-hop	in	—
	protocol	is、 in	—
	eigrp-metric	—	add、 set
	tag	is、 eq、 ge、 le	set
redistribute	destination	in	—
	next-hop	in	—
	mpls-label	route-has-label	—
	eigrp-metric	—	add、 set
	route-type	is	—
	tag	is、 eq、 ge、 le	set

RIP ポリシー接続点

この項では、それぞれの RIP ポリシー接続点について説明し、RIP 属性と演算子の概要を示します。

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 RIP アップデートに挿入できます。これは付加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 のルートがポリシーをパスする場合、デフォルト ルートが挿入されます。

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

router rip
  default-information originate route-policy rip-originate
```


Redistribute

RIP 内の `redistribute` 接続点を使用して、他のルーティングプロトコルソースから RIP データベースにルートを挿入できます。

次の例は、RIP に OSPF ルートを挿入する方法を示しています。

```
route-policy redist-ospf
  set rip-metric 5
end-policy

router rip
  redistribute ospf 1 route-policy redist-ospf
```

Global-Inbound

RIP 用 `global-inbound` 接続点を使用して、ルート ポリシーに一致するインバウンド RIP ルートのフィルタリングや更新が行えます。

次の例に、`rip-in` という名前のルート ポリシーに一致するインバウンド RIP ルートをフィルタする方法を示します。

```
router rip
  route-policy rip-in in
```

Global-Outbound

RIP 用 `global-outbound` 接続点を使用して、ルート ポリシーに一致するアウトバウンド RIP ルートのフィルタリングや更新が行えます。

次の例に、`rip-out` という名前のルート ポリシーに一致するアウトバウンド RIP ルートをフィルタする方法を示します。

```
router rip
  route-policy rip-out out
```

Interface-Inbound

`interface-inbound` 接続点を使用して、特定のインターフェイス向けルート ポリシーに一致するインバウンド RIP ルートのフィルタリングや更新が行えます。

次の例に、インターフェイス `0/1/0/1` 向けルート ポリシーに一致するインバウンド RIP ルートをフィルタする方法を示します。

```
router rip
  interface GigabitEthernet0/1/0/1
  route-policy rip-in in
```

Interface-Outbound

interface-outbound 接続点を使用して、特定のインターフェイス向けルート ポリシーに一致するアウトバウンド RIP ルートのフィルタリングや更新が行えます。

次の例に、インターフェイス 0/2/0/1 向けルート ポリシーに一致するアウトバウンド RIP ルートをフィルタする方法を示します。

```
router rip
  interface GigabitEthernet0/2/0/1
    route-policy rip-out out
```

RIP 属性と演算子

この表では、接続点ごとの RIP 属性と演算子をまとめます。

表 7: RIP 属性と演算子

接続点	属性	一致	設定
default-information originate	next-hop	na	set
	rip-metric	na	set
	rip-tag	na	set
	rib-has-route	in	na
global-inbound	destination	in	na
	next-hop	in	na
	rip-metric	na	add
global-outbound	destination	in	na
	protocol	is、 in	na
	rip-metric	na	add
interface-inbound	destination	in	na
	next-hop	in	na
	rip-metric	na	add

接続点	属性	一致	設定
interface-outbound	destination	in	na
	protocol	is、 in	na
	rip-metric	na	add
redistribute	destination	in	na
	next-hop	in	set
	rip-metric	na	set
	rip-tag	na	set
	mpls-label	route-has-label	na
	route-type	is	na
	tag	is、 eq、 ge、 le	set

PIM ポリシー接続点

この項では、PIM ポリシー **rpf-topology** 接続点について説明し、PIM 属性と演算子の概要を示します。

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ある場合もあります。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す。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。未定義のセットやポリシーブロックを参照する設定を作成しておき、後から未定義のポリシーやセットを埋めることが可能です。この設定の定義方法を使用すれば、ポリシー定義はずっと柔軟になります。ポリシーを定義するときに参照するすべてのポリシーが、設定に存在してい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` ステートメントによってポリシー `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することも可能です。

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、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。Cisco IOS XR 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

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、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は、次の URL を参照してください。 <http://www.nano-editor.org/>

Cisco IOS XR 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。

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、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/RSP0
/CPU0:router# edit route-policy policy_A
-----
== MicroEMACS 3.8b () == rpl_edit.139281 ==
  if destination in (2001::/8) then
    drop
  endif
end-policy
!

== MicroEMACS 3.8b () == rpl_edit.139281 ==
Parsing.
83 bytes parsed in 1 sec (82)bytes/sec
Committing.
1 items committed in 1 sec (0)items/sec
Updating.
Updated Commit database in 1 sec

RP/0/RSP0
/CPU0:router#
```

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/RSP0
/CPU0:router#edit route-policy policy_B
== MicroEMACS 3.8b () == rpl_edit.141738
route-policy policy_B
  set metric-type type_1
  if destination in (2001::/8) then
    drop
  endif
end-policy
!
== MicroEMACS 3.8b () == rpl_edit.141738 ==
Parsing.
105 bytes parsed in 1 sec (103)bytes/sec

% Syntax/Authorization errors in one or more commands.!! CONFIGURATION
FAILED DUE TO SYNTAX/AUTHORIZATION ERRORS
  set metric-type type_1
  if destination in (2001::/8) then
    drop
  endif
end-policy
```

```
!
Continue editing? [no]:
```

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グローバル コンフィギュレーション モードに戻るには、**abort** コマンドを使用します。

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。

階層的な条件

階層的な条件を使用することで、別のルート ポリシーの「if」文内でルート ポリシーを指定できるようになっています。ルート ポリシーを階層的ポリシーに基づいてコンフィギュレーションに適用できます。

階層的な条件機能によって、Cisco IOS-XR RPL 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をサポートしています。

条件ポリシーの適用

Cisco IOS-XR RPL 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をサポートしています。条件ポリシーの適用によって、別のルートポリシーの「if」文内でルート ポリシーの使用が可能になります。

ChildA、ChildB および Parent というルート ポリシー コンフィギュレーションについて考えてみます。

```
route-policy ChildA
  if destination in (10.10.0.0/16) then
    set local-pref 111
  endif
end-policy
!
route-policy ChildB
  if as-path originates-from '222' then
    set community (333:222) additive
  endif
end-policy
!
route-policy Parent
  if apply ChildA and apply ChildB then
    set community (333:333) additive
  else
    set community (333:444) additive
  endif
end-policy
!
```

前記シナリオでは、ポリシー Parent が実行されるたびに、「Parent」ポリシーの「if」条件がポリシー ChildA と ChildB の結果に基づいて判定されます。ポリシー「Parent」は、次に示す「merged」ポリシーと同等になります。

```
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 ChildA is pass */
    set local-pref 111
    set community (333:444) additive /*From else block */
  elseif as-path originates-from '222' then /*Only Policy ChildB is pass */
    set community (333:222, 333:444) additive /*From else block */
  else
    set community (333:444) additive /*From else block */
  endif
end-policy
```

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、次の適用条件を含むよう拡張できます。

```
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
```

ネストされたワイルドカード適用ポリシー

Routing Policy Language (RPL) の階層構造により、あるポリシーから別のポリシーを参照できます。参照されたまたは呼び出されたポリシーは、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。呼び出しや親ポリシーは、BGP ネイバーの共通セットへのアタッチのため、複数の子ポリシーをネストできます。ネストされたワイ

ワイルドカード適用ポリシーによって、ワイルドカード (*) ベースの適用のネスティングが行えます。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` ステートメントのポリシー名末尾にアスタリスク (*) を付けることで指定します。ワイルドカードポリシーへのパラメータ渡し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どの値にもマッチすることを表します。

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。子ポリシーはそれぞれグローバルに定義されているため、親はポリシー名をもとに子ポリシーを直接ネストしています。親は1度設定され、必要に応じてそれぞれの子ポリシーを継承します。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。

ルーティング ポリシーの実装方法

ここでは、次の手順について説明します。

ルート ポリシーの定義

このタスクでは、ルート ポリシーの定義方法を説明します。



(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。

手順の概要

1. **configure**
2. **route-policy name** [*parameter1*, *parameter2*, ..., *parameterN*]
3. **end-policy**
4. 次のいずれかを実行します。
 - **end**
 - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	configure 例： RP/0/RSP0/CPU0:router# configure	グローバル 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	route-policy name [<i>parameter1</i> , <i>parameter2</i> , ..., <i>parameterN</i>] 例： RP/0/RSP0/CPU0: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。 • route policy と入力された後、一群のコマンドを入力してルート ポリシーを定義できます。
ステップ 3	end-policy 例： RP/0/RSP0/CPU0: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	次のいずれかを実行します。 • end • commit 例： RP/0/RSP0/CPU0:router (config)# end または RP/0/RSP0/CPU0:router (config)# commit	設定変更を保存します。 • end コマンドを実行すると、変更をコミットするように要求されます。 Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]: ◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。 ◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。

	コマンドまたはアクション	目的
		<p>° cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。</p> <p>実行コンフィギュレーションファイルに変更を保存し、コンフィギュレーションセッションを継続するには、commit コマンドを使用します。</p>

ルーティング ポリシーの BGP ネイバーへのアタッチ

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。

はじめる前に

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。

手順の概要

1. **configure**
2. **router bgp as-number**
3. **neighbor ip-address**
4. **address-family { ipv4 unicast | ipv4 multicast | ipv4 labeled-unicast | ipv4 tunnel | ipv4 mdt | ipv6 unicast | ipv6 multicast | ipv6 labeled-unicast | vpv4 unicast | vpv6 unicast }**
5. **route-policy policy-name { in | out }**
6. 次のいずれかを実行します。
 - **end**
 - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	configure 例： RP/0/RSP0/CPU0:router# configure	グローバル コンフィギュレーション モードを開始します。

	コマンドまたはアクション	目的
ステップ 2	<p>router bgp <i>as-number</i></p> <p>例 :</p> <pre>RP/0/RSP0/CPU0:router(config)# router bgp 125</pre>	<p>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。</p> <ul style="list-style-type: none"> • 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	<p>neighbor <i>ip-address</i></p> <p>例 :</p> <pre>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.20</pre>	<p>ネイバー IP アドレスを指定します。</p>
ステップ 4	<p>address-family { <i>ipv4 unicast</i> <i>ipv4 multicast</i> <i>ipv4 labeled-unicast</i> <i>ipv4 tunnel</i> <i>ipv4 mdt</i> <i>ipv6 unicast</i> <i>ipv6 multicast</i> <i>ipv6 labeled-unicast</i> <i>vpn4 unicast</i> <i>vpn6 unicast</i> }</p> <p>例 :</p> <pre>RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast</pre>	<p>アドレス ファミリを指定します。</p>
ステップ 5	<p>route-policy <i>policy-name</i> { in out }</p> <p>例 :</p> <pre>RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy example1 in</pre>	<p>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。</p>
ステップ 6	<p>次のいずれかを実行します。</p> <ul style="list-style-type: none"> • end • commit <p>例 :</p> <pre>RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# end</pre> <p>または</p> <pre>RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# commit</pre>	<p>設定変更を保存します。</p> <ul style="list-style-type: none"> • end コマンドを実行すると、変更をコミットするように要求されます。 <pre>Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:</pre> <ul style="list-style-type: none"> ◦ yes と入力すると、実行コンフィギュレーションファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。 ◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。 ◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィ

	コマンドまたはアクション	目的
		<p>ギョレーションセッションは終了せず、設定変更もコミットされません。</p> <p>実行コンフィギュレーションファイルに変更を保存し、コンフィギュレーションセッションを継続するには、commit コマンドを使用します。</p>

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。テキストエディタの詳細については、[ルーティング ポリシー設定要素の編集](#)、(68 ページ) を参照してください。

手順の概要

1. `edit { route-policy | prefix-set | as-path-set | community-set | extcommunity-set { rt | soo } | policy-global | rd-set } name [nano | emacs | vim | inline { add | prepend | remove } set-element]`
2. `show rpl route-policy [name [detail] | states | brief]`
3. `show rpl prefix-set [name | states | brief]`

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	<pre><code>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]</code></pre> <p>例 :</p> <pre><code>RP/0/RSP0/CPU0:router# edit route-policy sample1</code></pre>	<p>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。</p> <ul style="list-style-type: none"> • 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。

	コマンドまたはアクション	目的
		<ul style="list-style-type: none"> • 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 [<i>name</i> [detail] states brief] 例 : <pre>RP/0/RSP0/CPU0:router# show rpl route-policy sample2</pre>	(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。 <ul style="list-style-type: none"> • 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。 • コンフィギュレーションは表示せずに拡張コミュニティセットすべての名前をリストするには brief キーワードを使用します。
ステップ 3	show rpl prefix-set [<i>name</i> states brief] 例 : <pre>RP/0/RSP0/CPU0:router# show rpl prefix-set prefixset1</pre>	(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。 <ul style="list-style-type: none"> • 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。

ルーティング ポリシーの実装の設定例

ここでは、次の設定例について説明します。

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

```

シンプルインバウンド ポリシー : 例

次のポリシーは、そのネットワーク層到達可能性情報（NLRI）が/24 より長いプレフィックスを指定しているルート、およびNLRIが RFC 1918 で予約されたアドレス空間の宛先を指定しているルートをすべて廃棄します。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 リストに範囲 101:202 ~ 106:202 内の値を含み、値 202 を含む 16 ビット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 2 度先頭に追加し、コミュニティ 2:666 をルート内のリストに追加します。これらのルートについて、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と設定し、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

```

モジュール型インバウンドポリシー : 例

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。`set-lpref-prepend` ポリシーは、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の設定と AS パスの先頭への追加が行える、ユーティリティポリシーです。`common-inbound` ポリシーは、インバウンドポリシーの共通ブロック作成に `filter-bogons` 構築ブロックを使用します。`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。

これは、ポリシー言語のモジュール化機能を示すシンプルな例だといえます。

```
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

```

その他の参考資料

ここでは、RPL の実装に関する関連資料について説明します。

関連資料

関連項目	参照先
ルーティングポリシー言語コマンド：コマンド構文の詳細、コマンドモード、コマンド履歴、デフォルト設定、使用に関する注意事項、および例	『Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference』の「Routing Policy Language Commands on Cisco ASR 9000 シリーズ ルータ」モジュール
正規表現の構文	『Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide』の付録「Understanding Regular Expressions, Special Characters and Patterns」

標準

標準	タイトル
この機能でサポートされる新規の標準または変更された標準はありません。また、既存の標準のサポートは変更されていません。	—

MIB

MIB	MIB のリンク
—	Cisco IOS XR ソフトウェアを使用して MIB の場所を特定してダウンロードするには、次の URL にある Cisco MIB Locator を使用して、[Cisco Access Products] メニューからプラットフォームを選択します。 http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFC

RFC	タイトル
RFC 1771	『A Border Gateway Protocol 4 (BGP-4)』
RFC 4360	『BGP Extended Communities Attribute』

シスコのテクニカル サポート

説明	リンク
シスコのテクニカルサポート Web サイトには、数千ページに及ぶ検索可能な技術情報があります。製品、テクノロジー、ソリューション、技術的なヒント、およびツールへのリンクもあります。Cisco.com に登録済みのユーザは、このページから詳細情報にアクセスできます。	http://www.cisco.com/techsupport

