



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 \(1 ページ\)](#)
- [ルートポリシーの定義 \(2 ページ\)](#)
- [BGP ネイバーへのルーティングポリシーのアタッチ \(4 ページ\)](#)
- [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 \(5 ページ\)](#)
- [ルーティングポリシーの参照 \(8 ページ\)](#)

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。
 - IPv4 ピアから始まるルートについては、RPL ポリシーを介してネクストホップアドレスを IPv6 アドレスに変更することはできません。

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
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

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要素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ります。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、コロンの区切った、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)
($ipaddr:666, 1234:$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/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（**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。

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 `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。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ず、**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。

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こ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
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 ファミリー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。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 1: 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 2: 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	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	該当なし	禁止	n/a	n/a
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スペ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 3: 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 4: 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。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、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 ==
```

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

```
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、XREXEC 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せん。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` を使用してこ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。