BGP 機能の概要

BGP はトランスポート プロトコルとして TCP を使用します。2 台の BGP ルータが互いの間に TCP 接続を形成し（ピア ルータ）、接続パラメータを開いて確認するためにメッセージを交換します。

BGP ルータはネットワーク到達可能性情報を交換します。この情報は、主に、宛先ネットワークに到達するためにルートで経由する必要のあるフル パス（BGP 自律システム番号）を示します。この情報は、ループ フリーである自律システムや、ルーティング動作に制限が適用されるルーティング ポリシーを表すグラフの作成に役立ちます。

TCP 接続を確立して BGP ルーティング情報を交換している 2 台のルータは、ピアまたはネイバーと呼ばれます。BGP ピアは最初に BGP ルーティング テーブル全体を交換します。この交換の後、ルーティング テーブルが変更されたとき差分更新が送信されます。BGP は BGP テーブルのバージョン番号を保存します。これはすべての BGP ピアで同一です。ルーティング情報の変更によって BGP がテーブルを更新するたびに、バージョン番号は変更されます。BGP ピア間の接続が維持されていることを確認するキープアライブ パケットが送信され、エラーまたは特殊な状態に応じて通知パケットが送信されます。

（注） OS XR Release 6.1.31 以降で VPNv4 アドレス ファミリがサポートされています。ただし、VPNv6 および VPN ルーティング/転送（VRF）アドレス ファミリは、今後のリリースでサポートされる予定です。

BGP ルーティングの有効化

BGP ルーティングをイネーブルにし、BGP ルーティング プロセスを設定するには、次の作業を実行します。BGP ネイバーの設定は、BGP ルーティングのイネーブル化の一部として含まれています。

（注）

- BGP ルーティングをイネーブルにするには、1 つ以上のネイバーおよび 1 つ以上のアドレス ファミリを設定する必要があります。**address family** コマンドおよび **remote as** コマンドを使用して、リモート AS とアドレス ファミリの両方を持つ 1 つ以上のネイバーをグローバルに設定する必要があります。
- 1 つの BGP セッションに IPv4 ユニキャストと IPv4 ラベル付きユニキャスト AFI/SAF の両方がある場合、ルーティング動作は非決定的になります。したがって、プレフィックスが正しくアドバタイズされない場合があります。プレフィックスが正しくアドバタイズされないと、到達可能性の問題が発生します。このような到達可能性の問題を回避するには、IPv4 ユニキャストまたは IPv4 ラベル付きユニキャスト アドレス ファミリのいずれかを介してプレフィックスをアドバタイズするルート ポリシーを明示的に設定する必要があります。

始める前に

BGP はルータ ID（設定済みループバック アドレスなど）を取得できなければなりません。1 つ以上のアドレス ファミリを BGP ルータ コンフィギュレーションに設定する必要があり、同じアドレス ファミリをネイバーの下にも設定する必要があります。

（注）ネイバーが外部 BGP（eBGP）ピアとして設定されている場合は、**route-policy** コマンドを使用して、インバウンドおよびアウトバウンドのルート ポリシーをネイバー上に設定する必要があります。

手順

ステップ 1 **configure**

ステップ 2 **route-policy** *route-policy-name*

例：

```
RP/0/RP0/cpu 0: router(config)# route-policy drop-as-1234
  RP/0/RP0/cpu 0: router(config-rpl)# if as-path passes-through '1234' then
  RP/0/RP0/cpu 0: router(config-rpl)# apply check-communities
  RP/0/RP0/cpu 0: router(config-rpl)# else
  RP/0/RP0/cpu 0: router(config-rpl)# pass
```

```
RP/0/RP0/cpu 0: router(config-rpl)# endif
```

（任意）ルート ポリシーを作成し、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。このモードではルート ポリシーを定義できます。

ステップ 3 **end-policy**

例：

```
RP/0/RP0/cpu 0: router(config-rpl)# end-policy
```

（任意）ルート ポリシーの定義を終了し、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**

ステップ 5 **configure**

ステップ 6 **router bgp** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config)# router bgp 120
```

BGP AS 番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 7 **bgp router-id** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# bgp router-id 192.168.70.24
```

指定したルータ ID で、ローカル ルータを設定します。

ステップ 8 **address-family** { **ipv4** | **ipv6** } **unicast**

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 9 **exit**

例：

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

現在のコンフィギュレーション モードを終了します。

ステップ 10 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.168.40.24
```

BGPルーティングのためにルータをネイバーコンフィギュレーションモードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

ステップ 11 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 2002
```

ネイバーを作成し、リモート自律システム番号を割り当てます。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 13 **route-policy** *route-policy-name* { **in** | **out** }

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# route-policy drop-as-1234 in
```

（任意）指定したポリシーを着信 IPv4 ユニキャスト ルートに適用します。

ステップ 14 **commit**

BGP のイネーブル化：例

次に、BGP をイネーブルにする例を示します。

```
prefix-set static
   2020::/64,
   2012::/64,
   10.10.0.0/16,
   10.2.0.0/24
end-set

route-policy pass-all
  pass
end-policy
route-policy set_next_hop_agg_v4
  set next-hop 10.0.0.1
end-policy
```

```
route-policy set_next_hop_static_v4
  if (destination in static) then
    set next-hop 10.1.0.1
  else
    drop
  endif
end-policy

route-policy set_next_hop_agg_v6
  set next-hop 2003::121
end-policy

route-policy set_next_hop_static_v6
  if (destination in static) then
     set next-hop 2011::121
  else
     drop
  endif
end-policy

router bgp 65000
  bgp fast-external-fallover disable
  bgp confederation peers
    65001
    65002
  bgp confederation identifier 1
  bgp router-id 1.1.1.1
  address-family ipv4 unicast
    aggregate-address 10.2.0.0/24 route-policy set_next_hop_agg_v4
    aggregate-address 10.3.0.0/24
    redistribute static route-policy set_next_hop_static_v4
  address-family ipv6 unicast
    aggregate-address 2012::/64 route-policy set_next_hop_agg_v6
    aggregate-address 2013::/64
    redistribute static route-policy set_next_hop_static_v6
    neighbor 10.0.101.60
    remote-as 65000
    address-family ipv4 unicast
    neighbor 10.0.101.61
    remote-as 65000
    address-family ipv4 unicast
    neighbor 10.0.101.62
    remote-as 3
    address-family ipv4 unicast
      route-policy pass-all in
      route-policy pass-all out
    neighbor 10.0.101.64
    remote-as 5
    update-source Loopback0
    address-family ipv4 unicast
      route-policy pass-all in
      route-policy pass-all out
```

BGP タイマーの調整

BGP は、定期実行アクティビティ（キープアライブ メッセージの送信、ネイバーがダウンしたと判断する条件となるそのネイバーからメッセージを受信しなかった期間など）を制御するために、特定のタイマーを使用します。ルータ コンフィギュレーション モードで timers bgp コ

マンドを使用して設定した値は、特定のネイバーでネイバー コンフィギュレーション モードで timers コマンドを使用すると上書きできます。

BGP ネイバーにタイマーを設定するには、次の作業を実行します。

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 123
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **timers bgp** *keepalive hold-time*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# timers bgp 30 90
```

すべてのネイバーのデフォルトのキープアライブ時間とデフォルトの保留時間を設定します。

ステップ 4 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.168.40.24
```

BGP ルーティングのためにルータをネイバー コンフィギュレーション モードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

ステップ 5 **timers** *keepalive hold-time*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# timers 60 220
```

（任意）BGP ネイバーのキープアライブ タイマーと保持時間タイマーを設定します。

ステップ 6 **commit**

BGP デフォルト ローカル プリファレンス値の変更

BGP パスのデフォルト ローカル プリファレンス値を設定するには、次の作業を実行します。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **bgp default local-preference** *value*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# bgp default local-preference 200
```

デフォルト値 100 以外のデフォルト ローカル プリファレンス値を設定します。100 より大きい値を設定して推奨度を上げるか、または 100 未満の値を設定して推奨度を低くすることができます。

ステップ 4 **commit**

BGP の MED メトリックの設定

メトリックがまだ設定されていないルート（MED 属性が設定されていない、受信されたルート）をピアにアドバタイズするように Multi Exit Discriminator（MED）を設定するには、次の作業を実行します。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **default-metric** *value*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# default metric 10
```

まだメトリックが設定されていないルート（MED 属性を持たない、受信されたルート）をピアにアドバタイズするように MED を設定する場合に使用されるデフォルトのメトリックを設定します。

ステップ 4 **commit**

BGP ウェイトの設定

重みとは、ベストパス選択プロセスを制御するためにパスに割り当てる数値です。ほとんどのトラフィックで特定のネイバーを優先する場合、**weight** コマンドを使用して、そのネイバーから学習したすべてのルートに大きい重みを割り当てることができます。ネイバーから受信しルートに重みを割り当てるには、次の作業を実行します。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.168.40.24
```

BGP ルーティングのためにルータをネイバー コンフィギュレーション モードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

ステップ 4 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 2002
```

ネイバーを作成し、リモート自律システム番号を割り当てます。

ステップ 5 **address-family** { **ipv4** | **ipv6** } **unicast**

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 6 **weight** *weight-value*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# weight 41150
```

ネイバーから学習したすべてのルートに重みを割り当てます。

ステップ 7 **commit**

次のタスク

新たに設定したウェイトを反映するには、clear bgp コマンドを使用します。

BGP 最適パス計算の調整

BGP ルータは、通常は同じ宛先に対する複数のパスを受信します。BGP の最適パス アルゴリズムは、IP ルーティング テーブルに格納し、トラフィックの転送に使用する最適なパスを決めるものです。BGP 最適パスは、次の 3 つのステップで構成されます。

- ステップ 1：2 つのパスを比較して、いずれが優れているのかを判別します。
- ステップ 2：すべてのパスを順に処理し、全体として最適なパスを選択するためにパスを比較する順序を決定します。
- ステップ 3：新しい最適パスを使用するに足るだけの差が新旧の最適パスにあるかどうかを判別します。

（注） 比較演算が推移的ではないため、ステップ 2 で決定された比較の順序は重要です。つまり、3 つのパス、A、B、C がある場合、A と B を比較したときに A の方が優れていて、B と C と比較したときに B の方が優れている場合、A と C を比較したときに必ずしも A が優れているとは限りません。この非推移性は、Multi Exit Discriminator（MED）がすべてのパス間ではなく、同じネイバー自律システム（AS）からのパス間のみで比較されるために生じます。BGP 最適パス アルゴリズム （137 ページ）で概念的な詳細を提供します。

デフォルトの BGP 最適パスの計算の動作を変更するには、次の作業を実行します。

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 126
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **bgp bestpath med missing-as-worst**

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# bgp bestpath med missing-as-worst
```

このパスを最も必要のないパスにするために、このパス内の不明 MED 属性の値は無限であると見なすように、BGP ソフトウェアに指示します。

ステップ 4 **bgp bestpath med always**

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# bgp bestpath med always
```

パスがどの自律システムから受信されたかに関係なく、すべてのパスの間でプレフィックスについて MED を比較するように、指定した自律システムの BGP スピーカーを設定します。

ステップ 5 **bgp bestpath med confed**

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# bgp bestpath med confed
```

コンフェデレーション ピアから学習したパスについて MED 値を BGP ソフトウェアで比較できるようにします。

ステップ 6 **bgp bestpath as-path ignore**

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# bgp bestpath as-path ignore
```

最適パスを選択するときに、自律システム パスの長さが無視されるように BGP ソフトウェアを設定します。

ステップ 7 **bgp bestpath compare-routerid**

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# bgp bestpath compare-routerid
```

類似パスのルータ ID を比較するように自律システムの BGP スピーカーを設定します。

ステップ 8 **commit**

BGP アドミニストレーティブ ディスタンスの設定

アドミニストレーティブディスタンスは、ルーティング情報源の信頼性を示す評価基準です。通常は、値が大きいほど、信頼性の格付けが下がります。一般的にルートは複数のプロトコルによって検出されます。アドミニストレーティブディスタンスは、複数のプロトコルから学習したルートを区別するために使用されます。最もアドミニストレーティブディスタンスが低いルートが IP ルーティング テーブルに組み込まれます。BGP はデフォルトで、次に示すアドミニストレーティブ ディスタンスを使用します。

表 *1:* デフォルトの *BGP* アドミニストレーティブ ディスタンス

ディスタンス	デフォルト値	機能
外部	20	eBGP から学習したルートに適用されます。
内部	200	iBGP から学習したルートに適用されます。
ローカル	200	ルータを起点とするルートに適用されます。

（注） ディスタンスは BGP パス選択アルゴリズムに影響しませんが、BGP で学習されたルートを IP ルーティング テーブルに組み込むかどうかを左右します。

あるルートのクラスよりも別のルートのクラスを優先するために使用できるアドミニストレーティブ ディスタンスを使用することを指定するには、次の作業を実行します。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 4 **distance bgp** *external-distance internal-distance local-distance*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)# distance bgp 20 20 200
```

あるルートのクラスよりも別のルートのクラスを優先するために外部、内部、およびローカルのアドミニストレーティブディスタンスを設定します。値が高いほど、信頼性のランクは低くなります。

ステップ 5 **commit**

BGP バックドア ルートの指定

通常、eBGP を介して学習されたルートは、ディスタンスを理由として IP ルーティング テーブルに組み込まれます。ただし、2 つの AS には IGP-learned バックドア ルートと eBGP-learned のルートがあります。ポリシーは、IGP-learned パスを優先パスとして使用し、IGP パスが停止しているときに eBGP-learned パスを使用するなどの内容になります。

外部ボーダー ゲートウェイ プロトコル（eBGP）のアドミニストレーティブディスタンスに、ローカルにソースされた BGP ルートのアドミニストレーティブディスタンスを設定し、Interior Gateway Protocol（IGP）ルートよりも推奨度を低くするには、次の作業を実行します。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 4 **network** { *ip-address* / *prefix-length* | *ip-address mask* } **backdoor**

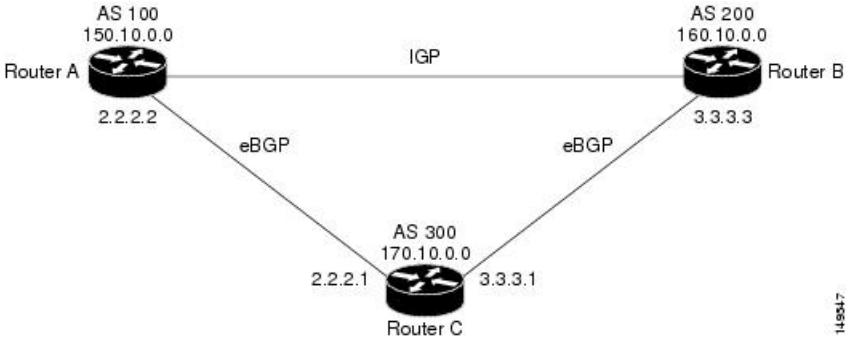
例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)# network 172.20.0.0/16
```

指定されたネットワークを作成してアドバタイズするようにローカル ルータを設定します。

ステップ 5 **commit**

バックドア：例



ここでは、ルータ A と C、ルータ B と C が eBGP を実行しています。ルータ A および B は、IGP を実行しています（ルーティング情報プロトコル（RIP）、Enhanced Interior Gateway Routing Protocol（IGRP）、Enhanced IGRP、または Open Shortest Path First（OSPF）など）。RIP、IGRP、Enhanced IGRP、および OSPF のデフォルト ディスタンスは、それぞれ、120、100、90、および 110 です。これらの距離はすべて eBGP のデフォルトディスタンス（20）よりも長くなります。通常は、ディスタンスの一番小さいルートが優先されます。

ルータ A は、160.10.0.0 に関するアップデートを、eBGP と IGP の 2 つのルーティング プロトコルから受信します。eBGP のデフォルトのディスタンスが IGP のデフォルトのディスタンスよりも低いので、ルータ A はルータ C からの eBGP-learned ルートを選択します。ルータ A にルータ B（IGP）からの 160.10.0.0 について学習させる場合は、BGP バック ドアを確立します。を参照してください。

次の例では、ネットワーク バックドアが設定されています。

```
RP/0/RP0/cpu 0: router(config)# router bgp 100
RP/0/RP0/cpu 0: router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/cpu 0: router(config-bgp-af)# network 160.10.0.0/16 backdoor
```

ルータ A では、eBGP-learned ルートをローカルとして扱い、ディスタンス 200 で IP ルーティング テーブルに組み込みます。このネットワークは Enhanced IGRP を介しても学習しているため（ディスタンスは 90）、Enhanced IGRP ルートは、IP ルーティング テーブルに正常に組み込まれ、トラフィックの転送に使用されます。Enhanced IGRP-learned ルートが停止すると、eBGP-learned ルートが IP ルーティング テーブルに組み込まれ、トラフィックの転送に使用されます。

Although BGP ではネットワーク 160.10.0.0 をローカル エントリとして扱いますが、通常、ローカル エントリをアドバタイズするようにネットワーク 160.10.0.0 をアドバタイズすることはありません。

集約アドレスの設定

BGP ルーティング テーブルに集約エントリを作成するには、次の作業を実行します。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 4 **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)# aggregate-address 10.0.0.0/8 as-set
```

集約アドレスを作成します。このルートにアドバタイズされたパスは、集約されるすべてのパスに含まれるすべての要素で構成された自律システム セットです。

- **as-set** キーワードは、関係するパスから自律システム セット パス情報およびコミュニティ情報を生成します。
- **as-confed-set** キーワードは、関係するパスから自律システム コンフェデレーション セット パス情報を生成します。
- **summary-only** キーワードは、アップデートから具体的なルートをすべてフィルタリングします。
- **route-policy** *route-policy-name* キーワードおよび引数は、集約ルートの属性の設定に使用されるルート ポリシーを指定します。

ステップ 5 **commit**

BGP MD5 認証の概要

BGP は、Message Digest 5（MD5）認証というメカニズムを、クリア テキストまたは暗号化されたパスワードを使用して 2 つの BGP ピア間での TCP セグメントの認証に提供します。

MD5 認証は BGP ネイバー レベルで設定します。MD5 認証を使用する BGP ピアは同じパスワードで設定します。パスワード認証に失敗した場合、パケットはセグメントに従って転送されません。

BGP MD5 認証の設定

2 つの BGP ピア間で BGP MD5 認証を設定するのには、この項の設定を使用します。

（注） MD5 認証の設定は、両方のピアとも同じです。

設定

BGP MD5 を設定するには、次の設定を使用します。

```
RP/0/RP0/cpu 0: router(config)# router bgp 50
RP/0/RP0/cpu 0: router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/cpu 0: router(config-bgp-af)# exit
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 10.1.1.1
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 51
RP/0/RP0/cpu 0: router(config-bgp-nbr)# password encrypted a1b2c3
RP/0/RP0/cpu 0: router(config-bgp-nbr)# commit
```

実行コンフィギュレーション

設定を検証します。

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

```
!
router bgp 50
address-family ipv4 unicast
!
neighbor 10.1.1.1
remote-as 51
password encrypted a1b2c3
!
!
```

BGP ネットワークのローカル AS 番号を非表示にする

2つの個別のBGPネットワークを単一のネットワークで組み合わせる場合は、自律システム番号を変更する必要があります。2 つのローカル自律システム番号をサポートして 2 つの BGP ネットワーク間のピアリングを維持するには、neighbor **local-as** コマンドを使用して BGP ピアを設定します。

ただし、neighbor **local-as** コマンドを BGP ピアに設定すると、デフォルトで eBGP ピアから学習したすべてのルートの前にローカル AS 番号が自動的に追加されます。ただし、この動作は、サービス プロバイダーや大規模な BGP ネットワークの自律システム番号の変更を困難にします。これは、付加されたAS番号付きのルートが、そのASに属している内部BGP（iBGP）ピアによって拒否されるためです。

no-prepend コマンドを使用してローカル AS 番号を非表示にすると、Border Gateway Protocol（BGP）ネットワークでの自律システム番号の変更プロセスが簡単になります。この機能を使用しないと、内部BGP（iBGP）ピアは、ルーティング ループを防止する as-path 属性内のローカル AS 番号を持つピアからの外部ルートを拒否します。ローカル AS 番号を非表示にすることで、BGPネットワーク全体の自律システム番号を透過的に変更でき、自律システムを通じてルートが伝達できるようにする一方で、AS 番号の遷移は不完全になります。

ローカル AS 番号を非表示にする BGP の設定

no-prepend コマンドを使用して eBGP ピアのローカル AS 番号を非表示にすると、BGP ネットワークの AS 番号を透過的に変更するのに使用でき、遷移時に AS 全体にルートが伝達されるようになります。ローカル AS 番号はこれらのルートに付加されないため、ある AS 番号から別の AS 番号への遷移時に内部ピアによって外部ルートが拒否されることはありません。

この項では、この機能の設定と確認について説明します。

（注） BGP は、ルートを通過する各 BGP ネットワークの自律システム番号を前に付加します。この動作は、ネットワーク到達可能性情報を維持してルーティング ループの発生を防ぐように設計されています。**no-prepend** コマンドを正しく設定しないと、ルーティング ループが発生します。そのため、このコマンドの設定は、経験豊富なネットワーク オペレータのみが行うようにしてください。

設定

次の設定を使用して、eBGP ピアのローカル AS 番号を非表示にします。

```
RP/0/RP0/cpu 0: router# config
RP/0/RP0/cpu 0: router(config)# router bgp 100
RP/0/RP0/cpu 0: router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/cpu 0: router(config-bgp-af)# network 10.1.1.1 255.255.0.0
RP/0/RP0/cpu 0: router(config-bgp-af)# neighbor 10.1.1.1 remote-as 100
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 10.1.1.1 local-as 300 no-prepend
RP/0/RP0/cpu 0: router(config-bgp)# commit
```

実行コンフィギュレーション

```
RP/0/RP0/cpu 0: router# show running-configuration
...
!
router bgp 100
 address-family ipv4 unicast
 network 10.1.1.1 255.255.0.0
 neighbor 10.1.1.1 remote-as 100
 neighbor 10.1.1.1 local-as 300 no-prepend
!
```

確認

次のコマンドを使用して、設定を確認します。

```
RP/0/RP0/cpu 0: router# show ip bgp neighbors
BGP neighbor is 10.1.1.1, remote AS 100, local AS 300 no-prepend, external link
BGP version 4, remote router ID 10.1.1.1
BGP state = Established, up for 00:00:49
Last read 00:00:49, hold time is 180, keepalive interval is 60 seconds
Neighbor capabilities:
Route refresh: advertised and received(new)
Address family IPv4 Unicast: advertised and received
IPv4 MPLS Label capability:
Received 10 messages, 1 notifications, 0 in queue
Sent 10 messages, 0 notifications, 0 in queue
Default minimum time between advertisement runs is 30 seconds
```

BGP の自律システム番号形式

自律システム番号（ASN）は、自律システム（AS）を識別するために使用されるグローバルに一意な識別子であり、これにより、AS では、ネイバー AS との間で外部ルーティング情報を交換できるようになります。一意の ASN は、BGP ルーティングで使用するために各 AS に割り当てられます。BGP では、ASN を 2 バイトの番号および 4 バイトの番号としてエンコードします。

2 バイト自律システム番号形式

2 バイト ASN は asplain 表記で表されます。2 バイトの範囲は 1 ～ 65535 です。

4 バイト自律システム番号形式

2 バイト自律システム番号（ASN）がいつか枯渇するときに備えて、BGP では 4 バイト ASN をサポートしています。4 バイト ASN は、asplain 表記と asdot 表記の両方で表されます。

asplain 表記での 4 バイト ASN のバイトの範囲は 1 ～ 4294967295 です。AS は 4 バイトの 10 進数として表されます。4 バイト ASN の asplain 表現は draft-ietf-idr-as-representation-01.txt で定義されています。

asdot 形式の 4 バイト ASN の場合は、4 バイトの範囲は 1.0 ～ 65535.65535 で、次の形式になります。

high-order-16-bit-value-in-decimal . low-order-16-bit-value-in-decimal

BGP の 4 バイト ASN 機能は、4 バイト AS 番号をサポートしていない BGP スピーカーをまたがって、4 バイトをベースとする AS パス情報を伝播するために使用されます。ASN のサイズを 2 バイトから 4 バイトに拡張するための情報については、draft-ietf-idr-as4bytes-12.txt を参照してください。AS は 4 バイトの 10 進数として表されます。

as-format コマンド

as-format コマンドは、ASN 表記を asdot に設定します。**as-format** コマンドを設定していない場合のデフォルト値は asplain です。

BGP Multi-Instance および Multi-AS

Multi-AS BGP を使用すると、Multi-Instance BGP の各インスタンスに異なる AS 番号を設定できるようになります。Multi-Instance および Multi-AS BGP は次の機能を備えています。

- 共通ルーティング インフラストラクチャを使用して、複数のルータによって提供されるサービスを単一の IOS-XR ルータに統合するメカニズム。
- 異なる BGP インスタンスに異なる AF を設定することにより、AF の分離を実現するメカニズム。
- 複数のインスタンス間でピアリングセッション全体を分散させることによって、セッションのスケールを高めることができる手段。
- 個々のインスタンスに異なる BGP テーブルを伝送させることにより、プレフィックスのスケール（特に RR で）を高めることができるメカニズム。
- 特定の状況における BGP コンバージェンスの改善。
- NSR を含むすべての BGP 機能は、すべてのインスタンスに対応しています。
- ロードおよびコミット ルータ レベルの操作は、以前に確認または適用された構成上で実行できます。

制約事項

- ルータは最大 4 つの BGP インスタンスをサポートします。
- 各 BGP インスタンスには、固有の ルータ ID が必要です。
- 各 BGP インスタンスで設定できるアドレス ファミリは 1 つだけです（VPNv4、VPNv6 および RT 制約は複数の BGP インスタンスで設定できます）。
- IPv4/IPv6 ユニキャストは、IPv4/IPv6 ラベル付きユニキャストが設定されている同じ BGP インスタンス内にある必要があります。
- IPv4/IPv6 マルチキャストは、IPv4/IPv6 ユニキャストが設定されている同じ BGP インスタンス内にある必要があります。
- 単一の BGP インスタンスに対するすべての設定変更を同時にコミットすることができます。ただし、複数のインスタンスに対する設定変更は同時にコミットできません。
- 同じリモート ルータとのピアリング時に、BGP の update-source をすべてのインスタンスのデフォルト VRF で一意にすることが推奨されます。

特定の自律システムに対する複数の BGP インスタンスの設定

特定の自律システムに複数の BGP インスタンスを設定するには、次のタスクを実行します。単一の BGP インスタンスに対するすべての設定変更を同時にコミットすることができます。ただし、複数のインスタンスに対する設定変更は同時にコミットできません。

手順

ステップ 1 **configure**

ステップ 2 **router bgp** *as-number* [**instance** *instance name*]

例：

```
RP/0/RSP0/CPU0:router(config)# router bgp 100 instance inst1
```

ユーザが指定した BGP インスタンスに対し BGP コンフィギュレーション モードを開始します。

ステップ 3 **bgp router-id***ip-address*

例：

```
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 10.0.0.0
```

BGP スピーキング ルータの固定ルータ ID（BGP インスタンス）を設定します。

（注） 各 BGP インスタンスに一意のルータ ID を手動で設定する必要があります。

ステップ 4 **commit**

BGP ルーティング ドメイン コンフェデレーション

iBGP メッシュを削減する方法の 1 つとして、ある自律システムを複数の副自律システムに分割し、単一のコンフェデレーションにグループ化することがあげられます。外部からは、このコンフェデレーションは単一の自律システムであるかのように見えます。各自律システムは内部で完全にメッシュ化されていて、同じコンフェデレーション内の他の自律システムとの間には数本の接続があります。異なる自律システム内にあるピアは eBGP セッションを持ちますが、ルーティング情報は iBGP ピアと同様な方法で交換されます。具体的には、ネクスト ホップ、MED、およびローカル プリファレンス情報は維持されます。この機能により、自律システムすべてに対して単一の IGP を保持できます。

BGP のルーティング ドメイン コンフェデレーションの設定

BGP のルーティング ドメイン コンフェデレーションを設定するには、次の作業を実行します。これには、コンフェデレーション ID の指定と、コンフェデレーションに属す自律システムの指定を含みます。

ルーティング ドメイン コンフェデレーションを設定すると、自律システムを複数の自律システムに分割して、これを 1 つのコンフェデレーションにグループ化することによって、内部 BGP（iBGP）メッシュを削減することができます。それぞれの自律システムは、そのシステム自身内で完全にメッシュ化されていて、同じコンフェデレーションの別の自律システムとの接続を数個持ちます。このコンフェデレーションによりネクストホップおよびローカルプリファレンス情報が維持され、これにより、すべての自律システムに対して Interior Gateway Protocol（IGP）を 1 つ維持できるようになります。外部からは、このコンフェデレーションは単一の自律システムであるかのように見えます。

手順

ステップ 1 **configure**

ステップ 2 **router bgp** *as-number*

例：

```
RP/0/RP0/cpu 0: router# router bgp 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **bgp confederation identifier** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# bgp confederation identifier 5
```

BGP コンフェデレーション ID を指定します。

ステップ 4 **bgp confederation peers** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# bgp confederation peers 1091
  RP/0/RP0/cpu 0: router(config-bgp)# bgp confederation peers 1092
  RP/0/RP0/cpu 0: router(config-bgp)# bgp confederation peers 1093
  RP/0/RP0/cpu 0: router(config-bgp)# bgp confederation peers 1094
  RP/0/RP0/cpu 0: router(config-bgp)# bgp confederation peers 1095
  RP/0/RP0/cpu 0: router(config-bgp)# bgp confederation peers 1096
```

BGP 自律システムが指定された BGP コンフェデレーション ID に属することを指定します。例に示すように、複数の AS 番号を同じコンフェデレーション ID に関連付けることができます。

ステップ 5 **commit**

BGP コンフェデレーション：例

次に、コンフェデレーションのいくつかのピアを表示する設定の例を示します。このコンフェデレーションは、自律システム番号 6001、6002、および 6003 の 3 つの内部自律システムから構成されています。コンフェデレーション外の BGP スピーカーには、このコンフェデレーションは（**bgp confederation identifier** コマンドによって指定される）自律システム番号 666 を持つ通常の自律システムのように見えます。

自律システム 6001 の BGP スピーカーで、**bgp confederation peers** コマンドは、自律システム 6002 および 6003 からのピアを特別な eBGP ピアとしてマークします。したがって、ピア 171.16 .232.55 および 171.16 .232.56 は、このアップデートでローカル プリファレンス、ネクスト ホップ、および未変更の MED を取得します。171 .19 .69.1 のルータは通常の eBGP スピーカーであり、このピアからのアップデートは、自律システム 666 のピアから受け取る通常の eBGP アップデートとまったく同じです。

```
router bgp 6001
 bgp confederation identifier 666
 bgp confederation peers
  6002
  6003
   exit
 address-family ipv4 unicast
  neighbor 171.16.232.55
  remote-as 6002
   exit
 address-family ipv4 unicast
  neighbor 171.16.232.56
  remote-as 6003
   exit
 address-family ipv4 unicast
  neighbor 171.19.69.1
  remote-as 777
```

自律システム 6002 の BGP スピーカーでは、自律システム 6001 および 6003 からのピアは特別な eBGP ピアとして設定されます。ピア 171 .17 .70.1 は通常の iBGP ピアであり、ピア 199.99.99.2 は自律システム 700 の通常の eBGP ピアです。

```
router bgp 6002
 bgp confederation identifier 666
 bgp confederation peers
  6001
  6003
   exit
 address-family ipv4 unicast
  neighbor 171.17.70.1
   remote-as 6002
   exit
 address-family ipv4 unicast
  neighbor 171.19.232.57
   remote-as 6001
   exit
 address-family ipv4 unicast
  neighbor 171.19.232.56
   remote-as 6003
   exit
 address-family ipv4 unicast
  neighbor 171.19.99.2
   remote-as 700
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
```

自律システム 6003 の BGP スピーカーでは、自律システム 6001 および 6002 からのピアは特別な eBGP ピアとして設定されます。ピア 192 .168 .200.200 は、自律システム 701 の通常の eBGP ピアです。

```
router bgp 6003
 bgp confederation identifier 666
 bgp confederation peers
  6001
  6002
   exit
 address-family ipv4 unicast
  neighbor 171.19.232.57
   remote-as 6001
   exit
 address-family ipv4 unicast
  neighbor 171.19.232.55
   remote-as 6002
   exit
 address-family ipv4 unicast
  neighbor 192.168.200.200
   remote-as 701
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
```

下記は、同じ例の自律システム 701 の BGP スピーカー 192 .168 .200.205 から受信する設定の一部です。ネイバー 171.16 .232.56 は自律システム 666 の通常の eBGP スピーカーとして設定されます。コンフェデレーション外部のピアは、この自律システムが複数の自律システムに内部分割されることを認識しません。

```
router bgp 701
 address-family ipv4 unicast
  neighbor 172.16.232.56
   remote-as 666
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
   exit
 address-family ipv4 unicast
  neighbor 192.168.200.205
   remote-as 701
```

BGP の追加パス

ボーダーゲートウェイプロトコル（BGP）の追加パス機能では、1つの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、BGP スピーカーの BGP プロトコル機械を変更します。これにより、ネットワークに「パスの多様性」が生まれます。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。

BGP 追加パスでは、iBGP ネットワーク内の追加パス アドバタイズメントが可能になり、プレフィックスに対する次のタイプのパスがアドバタイズされます。

- バックアップ パス：高速コンバージェンスおよび接続の回復をイネーブルにします。
- グループ最適パス：ルート振動を解決します。
- すべてのパス：iBGP フル メッシュをエミュレートします。

BGP 追加パスの設定

BGP 追加パス機能を設定するには、次の作業を行います。

手順

ステップ 1 **configure**

ステップ 2 **route-policy** *route-policy-name*

例：

```
RP/0/RP0/cpu 0: router (config)#route-policy add_path_policy
```

ルート ポリシーを定義して、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。

ステップ 3 **if** *conditional-expression* **then** *action-statement* **else**

例：

```
RP/0/RP0/cpu 0: router (config-rpl)#if community matches-any (*) then
    set path-selection all advertise
  else
```

特定のルートのアクションとディスポジションを決定します。

ステップ 4 **pass endif**

例：

```
RP/0/RP0/cpu 0: router(config-rpl-else)#pass
RP/0/RP0/cpu 0: router(config-rpl-else)#endif
```

処理のためにルートを渡し、if ステートメントを終了します。

ステップ 5 **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 6 **router bgp** *as-number*

例：

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

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 7 **address-family** {**ipv4** {**unicast** } | **ipv6** {**unicast** | **l2vpn vpls-vpws** | **vpnv4 unicast** | **vpnv6 unicast** }

例：

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

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

ステップ 8 **additional-paths receive**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)#additional-paths receive
```

対応ピアのプレフィックスのマルチパス受信機能を設定します。

ステップ 9 **additional-paths send**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)#additional-paths send
```

対応ピアのプレフィックスのマルチパス送信機能を設定します。

ステップ 10 **additional-paths selection route-policy** *route-policy-name*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)#additional-paths selection route-policy
add_path_policy
```

プレフィックスの追加パス選択機能を設定します。

ステップ 11 **commit**

BGP 最大プレフィックス

最大プレフィックス機能では、特定のアドレス ファミリのネイバーから受信されるプレフィックスの数に上限が課されます。受信されるプレフィックスの数が設定した最大数を超えると、停止通知がネイバーに送信された後、BGPセッションが終了します（これはデフォルト動作です）。手動によるクリアがユーザによって実行されるまで、セッションはダウンしたままになります。セッションは、**clear bgp** コマンドを使用して再開できます。**restart** キーワードを指定した **maximum-prefix** コマンドを使用して、セッションが自動的に起動されるまでの期間を設定できます。プレフィックスの上限はユーザが設定できます。ユーザがそのアドレス ファミリに対するプレフィックスの最大数を設定していない場合は、デフォルトの制限値が使用されます。

同じ回線で、最大プレフィックス値が変更された場合のアクションを次に示します。

- 最大値が単独で変更されると、必要に応じてルート更新メッセージが送信されます。
- 新しい最大値が現在のプレフィックス カウント ステートよりも大きい場合、新しいプレフィックス ステートが保存されます。
- 新しい最大値が現在のプレフィックス カウント ステートより小さい場合、新しく設定されたステートの値に一致するように、既存のプレフィックスが一部削除されます。

どのプレフィックスを削除するかを制御する方法は現在ありません。

過剰パスの破棄の設定

最大プレフィックス設定での過剰パスの破棄オプションを使用すると、プレフィックスが設定した最大値を超えた場合に、ネイバーから受信された過剰なプレフィックスをすべて廃棄できます。ただし、この廃棄によってセッション フラップが発生することはありません。

過剰パスの破棄オプションの利点は次のとおりです。

- BGP のメモリ フットスタンプが制限されます。
- パスが設定された制限を超えるとピアのフラッピングが停止します。

過剰パスの破棄設定が削除されると、BGP は更新機能をサポートしている場合にルート更新メッセージをネイバーに送信します。それ以外の場合、セッションはフラップします。

（注）
- ルータがプレフィックスを廃棄すると、ネットワークの残りと一致せず、ルーティングループが起きる可能性があります。
- プレフィックスが廃棄されると、スタンバイおよびアクティブ状態の BGP セッションが別のプレフィックスを廃棄する可能性があります。その結果、NSR スイッチオーバーによって BGP テーブルの矛盾が生じます。
- 過剰パスの破棄設定は、ソフト再設定構成と共存できません。

BGP 最大プレフィックス過剰パスの破棄を設定するには、次のタスクを実行します。

手順

ステップ 1 **configure**

例：
```
RP/0/RP0/cpu 0: router# configure
```
XR 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 **router bgp** *as-number*

例：
```
RP/0/RP0/cpu 0: router(config)# router bgp 10
```
自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **neighbor** *ip-address*

例：
```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 10.0.0.1
```
BGPルーティングのためにルータをネイバーコンフィギュレーションモードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

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

例：
```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# address-family ipv4 unicast
```
IPv4 または IPv6 のいずれかのアドレス ファミリを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

ステップ 5 **maximum-prefix** *maximum* **discard-extra-paths**

例：
```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# maximum-prefix 1000 discard-extra-paths
```
許可されるプレフィックス数の制限を設定します。

最大プレフィックスの制限を超えると過剰パスを破棄するように過剰パスの破棄を設定します。

ステップ 6 **commit**

例

次に、IPv4アドレス ファミリに対する過剰パスの破棄機能を設定する例を示します。

```
RP/0//CPU0:router# configure
RP/0//CPU0:router(config)# router bgp 10
RP/0//CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0//CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0//CPU0:router(config-bgp-nbr-af)# maximum-prefix 1000 discard-extra-paths
RP/0//CPU0:router(config-bgp-vrf-af)# commit
```

次の画面出力では、過剰パスの破棄オプションの詳細を示しています。

```
RP/0//CPU0:ios# show bgp neighbor 10.0.0.1

BGP neighbor is 10.0.0.1
Remote AS 10, local AS 10, internal link
Remote router ID 0.0.0.0
BGP state = Idle (No best local address found)
Last read 00:00:00, Last read before reset 00:00:00
Hold time is 180, keepalive interval is 60 seconds
Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
Last write 00:00:00, attempted 0, written 0
Second last write 00:00:00, attempted 0, written 0
Last write before reset 00:00:00, attempted 0, written 0
Second last write before reset 00:00:00, attempted 0, written 0
Last write pulse rcvd not set last full not set pulse count 0
Last write pulse rcvd before reset 00:00:00
Socket not armed for io, not armed for read, not armed for write
Last write thread event before reset 00:00:00, second last 00:00:00
Last KA expiry before reset 00:00:00, second last 00:00:00
Last KA error before reset 00:00:00, KA not sent 00:00:00
Last KA start before reset 00:00:00, second last 00:00:00
Precedence: internet
Multi-protocol capability not received
Received 0 messages, 0 notifications, 0 in queue
Sent 0 messages, 0 notifications, 0 in queue
Minimum time between advertisement runs is 0 secs

For Address Family: IPv4 Unicast
BGP neighbor version 0
Update group: 0.1 Filter-group: 0.0 No Refresh request being processed
Route refresh request: received 0, sent 0
0 accepted prefixes, 0 are bestpaths
Cumulative no. of prefixes denied: 0.
Prefix advertised 0, suppressed 0, withdrawn 0
Maximum prefixes allowed 10 (discard-extra-paths) <<<<<<<<<<<<<<<<<<<<<<
Threshold for warning message 75%, restart interval 0 min
AIGP is enabled
An EoR was not received during read-only mode
Last ack version 1, Last synced ack version 0
Outstanding version objects: current 0, max 0
```

```
Additional-paths operation: None
Send Multicast Attributes

Connections established 0; dropped 0
Local host: 0.0.0.0, Local port: 0, IF Handle: 0x00000000
Foreign host: 10.0.0.1, Foreign port: 0
Last reset 00:00:00
```

BGP の最適外部パス

最適外部パス機能では、ローカルで選択された最適パスが内部ピアからのパスの場合における、iBGP およびルート リフレクタ ピアへの最適外部パスのアドバタイズメントをサポートしています。BGP では各宛先に対して最適パスを 1 つとバックアップ パスを 1 つ選択します。デフォルトでは、最適パスを 1 つ選択します。さらに、BGP では、1 つのプレフィックスに対する残りの外部パスのうちから別の最適パスを選択します。1 つのパスのみが最適外部パスとして選択され、バックアップ パスとして他の PE に送信されます。BGP では、最適パスが iBGP パスの場合のみ最適外部パスを計算します。最適パスが eBGP パスの場合、最適外部パス計算は不要です。

最適外部パスを決定する手順を次に示します。

1. プレフィックスに利用可能なパスの全セットから最適パスを決定します。
2. 現在の最適パスを除外します。
3. このプレフィックスのすべての内部パスを除外します。
4. 残りのパスから、現在の最適パスと同じネクスト ホップを持つすべてのパスを除外します。
5. 残りのパスのセットに対して最適パス アルゴリズムを再度実行し、最適外部パスを決定します。

BGP では、1 つのプレフィックスに対する外部およびコンフェデレーションの BGP パスを考慮して最適外部パスを計算します。BGP では、最適パスおよび最適外部パスを次のようにアドバタイズします。

- プライマリ PE 上：プレフィックスの最適パスを内部と外部の両方のピアにアドバタイズ
- バックアップ PE 上：あるプレフィックスに対して選択された最適パスを外部ピアにアドバタイズし、このプレフィックスに対して選択された最適外部パスを内部ピアにアドバタイズ

最適外部パス アドバタイズメントの設定

iBGP およびルートリフレクタ ピアに最適外部パスをアドバタイズするには、次の作業を実行します。

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 100
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 次のいずれかを実行します。

- **address-family { vpnv4 unicast | vpnv6 unicast }**
- **vrf***vrf-name***{ipv4 unicast|ipv6 unicast}**

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# address-family vpnv4 unicast
```

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

ステップ 4 **advertise best-external**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)# advertise best-external
```

iBGP およびルートリフレクタ ピアに最適外部パスをアドバタイズします。

ステップ 5 **commit**

BGP Local Label Retention

プライマリ PE-CE リンクが故障した場合、BGP では、プライマリ パスに対応するルートおよびこのルートのローカル ラベルを取り消し、デフォルトでは、ルーティング情報ベース（RIB）および転送情報ベース（FIB）にバックアップ パスをプログラムします。

ただし、プライマリ PE のすべての内部ピアがバックアップ パスを新しい最適パスとして使用するように再コンバージェンスするまで、トラフィックは、プライマリ パスに割り当てられたローカル ラベルとともに、引き続きプライマリ PE に転送されます。したがって、プライマリ パスに前に割り当てられていたローカル ラベルは、再コンバージェンス後、設定可能な期間、プライマリ PE 上で保持する必要があります。BGP Local Label Retention 機能を使用すると、ローカル ラベルを指定期間保持できます。時間を指定していない場合、ローカル ラベルは、デフォルト値の 5 分間保持されます。

プライマリ パスのローカル ラベル割り当ての保持

プライマリ PE で以前にプライマリ パスに割り当てられたローカル ラベルを、再コンバージェンス後に設定期間にわたって保持するには、次の作業を実行します。

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 100
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **address-family { vpnv4 unicast | vpnv6 unicast }**

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# address-family vpnv4 unicast
```

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

ステップ 4 **retain local-label** *minutes*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)# retain local-label 10
```

プライマリ PE で以前にプライマリ パスに割り当てられたローカル ラベルを、再コンバージェンス後 10 分間保持します。

ステップ 5 **commit**

ローカル ラベル割り当ての保持：例

次に、プライマリ PE のプライマリ パスに以前に割り当てたローカル ラベルを再コンバージェンス後 10 分にわたって維持する例を示します。

```
router bgp 100
address-family l2vpn vpls-vpws
  retain local-label 10
end
```

BGP ラベル付きユニキャスト マルチ ラベル スタックの概要

BGP ラベル付きユニキャスト マルチ ラベル スタック機能では、ユーザがエンコードされたプレフィックスに関連付けられた 1 つ以上のラベルのスタックで BGP LU アップデートを XR ルータで受信しアドバタイズできます。

この機能は、マルチ ラベル スタックを BGP ラベル付きユニキャスト セッションを通じてコントローラがヘッドエンドにプッシュできるようにします。

前提条件

BGP ラベル付きユニキャスト アドレス ファミリがサポートされている必要があります。

制約事項

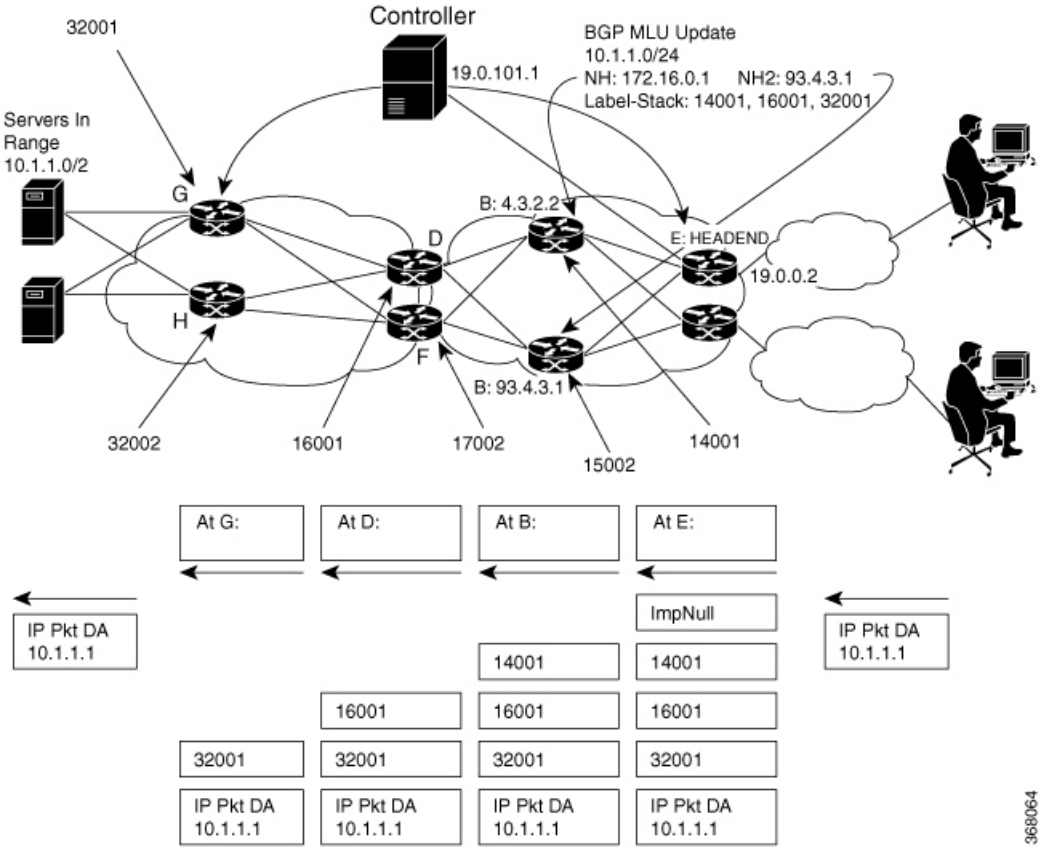
ハードウェアの制限により、最大 3 つのラベル スタックのみがサポートされます。リリース 6.6.1 以降では最大 5 つのラベルがサポートされます。

トポロジ

次の項で、BGP ラベル付きユニキャスト マルチ ラベル スタック機能のトポロジを図で示します。

コントローラがヘッド エンド E でプッシュしたマルチラベル スタックに基づき、トラフィックはネットワークを通って進みます。このトポロジでは、コントローラがラベル スタック 14001、16001、および 32001 を NH 172.6.0.1 を使用してプッシュすると、トラフィックはノード B、D、および G を順次通過して進みます。トラフィック パスをコントローラが連続してノード C、F、G に変更する必要がある場合、ラベル スタック 15002、17002、および 32001 を NH 93.4.3.1 を使用してプッシュします。

図 *1 : BGP* ラベル付きユニキャスト マルチ ラベル スタックのトポロジ



設定

この項では、BGP ラベル付きユニキャスト マルチ ラベル スタック機能の設定方法について説明します。

BGP コンフィギュレーション モードで **nexthop mpls forwarding ibgp** コマンドを設定します。BGP ラベル付きユニキャスト セッションをネクストホップ 10.3.2.2 で設定して、「ImpNULL」ラベルを最初のラベルとして複数ラベルスタックにプッシュします。

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# neighbor 10.0.1.101
Router(config-bgp)# nexthop mpls forwarding ibgp
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# allocate-label all
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.3.2.2
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family ipv4 labeled-unicast
Router(config-bgp)# exit
Router(config-bgp)# neighbor-group group 1
Router(config-bgp-nbrgrp)# neighbor-group group 1
Router(config-bgp-nbrgrp)# remote-as 65535
```

```
Router(config-bgp-nbrgrp)# address-family ipv4 labeled-unicast
Router(config-bgp-nbrgrp-af)# route-policy pass in
Router(config-bgp-nbrgrp-af)# route-policy pass out
Router(config-bgp-nbrgrp-af)# enforce-multiple-labels
Router(config-bgp-nbrgrp-af)# exit
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# neighbor 10.0.1.101
Router(config-bgp-nbr)# use neighbor-group ipv4lu_ng1
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config-bgp)# neighbor 10.0.1.101
Router(config-bgp-nbr)# remote-as 65535
Router(config-bgp-nbr)# address-family ipv4 labeled-unicast
Router(config-bgp-nbr-af)# route-policy pass in
Router(config-bgp-nbr-af)# route-policy pass out
Router(config-bgp-nbr-af)# route-reflector-client
Router(config-bgp-nbr-af)# enforce-multiple-labels
```

実行コンフィギュレーション

```
router bgp 100
bgp router-id 10.0.1.101
nexthop mpls forwarding ibgp
address-family ipv4 unicast
  allocate-label all
!
 neighbor 10.3.2.2
  remote-as 100
  address-family ipv4 labeled-unicast
!
neighbor-group ipv4lu_ng1
  remote-as 100
  address-family ipv4 labeled-unicast
   route-policy pass in
   route-policy pass out
   enforce-multiple-labels

neighbor 10.0.1.101
  use neighbor-group ipv4lu_ng1
  !
!
neighbor 10.0.1.101
  remote-as 100
  address-family ipv4 labeled-unicast
   route-policy pass out
   route-policy pass in
   route-reflector-client
   enforce-multiple-labels
!
```

確認

次の項に示す show の出力に、BGP LU マルチ ラベル スタック機能の設定の詳細と、それらの設定のステータスが表示されます。

```
/* Verify the multiple label stack. */
```

```
Router# show bgp ipv4 labeled-unicast 10.1.1.1/32

...

10.3.2.2 from 10.0.1.101

      Received Label 14001 16001 32001

      Origin incomplete, metric 0, localpref 94, valid, internal, best, group-best

      Received Path ID 0, Local Path ID 0, version 42

      Community: 258:259 260:261 262:263 264:265

      Large Community: 1:2:3 5:6:7
...

/* Verify if the multiple label stack is enabled.*/
Router# show bgp neighbor 10.0.1.101

...

 For Address Family: IPv4 Labeled-unicast

  BGP neighbor version 177675

  Update group: 0.8 Filter-group: 0.4  No Refresh request being processed

  Route-Reflector Client

  Send Multicast Attributes

 Multiple label stack: Enabled


/* Verify that the multiple label stack is enabled. */
Router# show bgp ipv4 labeled-unicast update-group 0.8

Update group for IPv4 Labeled-unicast, index 0.8:

  Attributes:

    Neighbor sessions are IPv4

    Outbound policy: ibgp-rpl1

    Internal

    Common admin

    First neighbor AS: 100

    Send communities

    Send GSHUT community if originated

    Send extended communities

    Route Reflector Client

    4-byte AS capable

    Send AIGP
```

```
    Send multicast attributes

     Multiple label stack: Enabled

/* Verify that the multiple label stack is enabled. */
Router# show bgp labels

...

Status codes: s suppressed, d damped, h history, * valid, > best

              i - internal, r RIB-failure, S stale, N Nexthop-discard

Origin codes: i - IGP, e - EGP, ? - incomplete

   Network            Next Hop        Rcvd Label      Local Label

*>i10.1.1.1/32         10.3.2.2          14001 16001     24193

                                       32001

*>i1.2.2.2/32          10.4.3.1        15002 17002     24199

                                        32002

*>i1.3.3.3/32          10.3.2.2         14001 16001     24200

                                       32002

...

/* */

Router# show route 10.1.1.1/32 detail

Routing entry for 10.1.1.1/32

  Known via "bgp 100", distance 200, metric 476387081, [ei]-bgp, labeled unicast (3107)

...

 Routing Descriptor Blocks

    209.165.201.1, from 10.0.1.101

      Route metric is 476387081

      Labels: 0x36b1 0x3e81 0x7d01 (14001 16001 32001)

      Tunnel ID: None

      Binding Label: None

      Extended communities count: 0

      NHID:0x0(Ref:0)

      MPLS eid:0x1380b00000003


/*  Verify that the multiple label stack is enabled. */
```

```
Router# show cef 10.1.1.1/32 detail

10.1.1.1/32, version 251579, internal 0x5000001 0x0 (ptr 0xa0241200) [1], 0x0 (0xa03feab8),
 0xa08

(0x9fced2b0)

 ...

   via 10.3.2.2/32, 3 dependencies, recursive [flags 0x6000]

    path-idx 0 NHID 0x0 [0x9e873ca0 0x0]

    recursion-via-/32

    next hop 10.3.2.2/32 via 24192/0/21

     local label 24193

     next hop 10.3.2.2/32 Te0/0/0/0/1  labels imposed {ImplNull 14001 16001 32001}


/* Verify the maximum supported depth of the label stack. If the number of labels received
 exceeds the maximum
supported by the platform, the prefix is not downloaded to the RIB and hence routing
issues may occur. */

Router# show bgp ipv4 labeled-unicast process performance detail

...

Address Family: IPv4 Labeled-unicast

State: Normal mode.

BGP Table Version: 177675

Attribute download: Disabled

ASBR functionality enabled

Label retention timer value 5 mins

Soft Reconfig Entries: 367

Table bit-field size : 1 Chunk element size : 3

Maximum supported label-stack depth:

   For IPv4 Nexthop: 3

   For IPv6 Nexthop: 0

...
```

iBGP マルチパス ロード シェアリング

ローカル ポリシーが設定されていないボーダー ゲートウェイ プロトコル（BGP）対応ルータが複数のネットワーク層到達可能性情報（NLRI）を同じ宛先の内部 BGP（iBGP）から受信す

ると、このルータは 1 つの iBGP パスを最適パスとして選択します。この最適パスは、次にこのルータの IP ルーティング テーブルに組み込まれます。iBGP のマルチパス ロード シェアリング機能を使用すると、BGP 対応ルータでは、複数の iBGP パスを宛先への最適パスとして選択できます。この最適パスまたはマルチパスは、次にこのルータの IP ルーティング テーブルに組み込まれます。

iBGP マルチパス ロード シェアリングの参照 （150 ページ） で、その他詳細情報を提供します。

iBGP マルチパス ロード シェアリングの設定

iBGP マルチパス ロード シェアリングを設定するには、次の作業を実行します。

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 100
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

ステップ 4 **maximum-paths ibgp** *number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)# maximum-paths ibgp 30
```

ロード シェアリング用の iBGP パスの最大数を設定します。

ステップ 5 **commit**

iBGP マルチパス負荷共有設定：例

次に、負荷共有に 30 のパスが使用されている設定の例を示します。

```
router bgp 100
 address-family ipv4 multicast
  maximum-paths ibgp 30
```

```
 !
!
end
```

ルート ダンプニング

ルート ダンプニングは、インターネットワーク上でのフラッピング ルートの伝搬を最小限に抑える BGP 機能です。ルートの状態が使用可能、使用不可能、使用可能、使用不可能という具合に、繰り返し変化する場合、ルートはフラッピングと見なされます。

たとえば、自律システム 1、自律システム 2、および自律システム 3 の 3 つの BGP 自律システムがあるネットワークについて考えます。自律システム 1 のネットワーク A へのルートがフラッピングする（利用できなくなる）と仮定します。ルート ダンプニングがない状況では、自律システム 1 から自律システム 2 への eBGP ネイバーは、取り消しメッセージを自律システム 2 に送信します。次に自律システム 2 内の境界ルータは、取り消しメッセージを自律システム 3 に伝播します。ネットワーク A へのルートが再出現したとき、自律システム 1 は自律システム 2 に、自律システム 2 は自律システム 3 にアドバタイズメント メッセージを送信します。ネットワーク A へのルートが利用可能になったり不可になったりを繰り返す場合、取り消しメッセージおよびアドバタイズメント メッセージが多数送信されます。ルート フラッピングは、インターネットに接続されたインターネットワークでの問題です。インターネットのバックボーンでルートのフラッピングが生じると、通常、多くのルートに影響を与えるからです。

ルート ダンプニング機能は、次のようにしてフラッピングの問題を最小限に抑えます。ここでも、ネットワーク A へのルートがフラッピングしたと仮定します。（ルート ダンプニングがイネーブルになっている）自律システム 2 内のルータは、ネットワーク A にペナルティ 1000 を割り当てて、履歴状態に移行させます。自律システム 2 内のルータは、引き続きネイバーにルートのステータスをアドバタイズします。ペナルティは累積されます。ルート フラップが非常に頻繁に発生し、ペナルティが設定可能な抑制制限を超える場合は、フラップの発生回数に関係なく、ルータはネットワーク A へのルートのアドバタイズを停止します。このようにして、ルート ダンプニングが発生します。

ネットワーク A に課されたペナルティは再使用制限に達するまで減衰し、達すると同時にそのルートは再びアドバタイズされます。再使用制限の半分の時点で、ネットワーク A へのルートのダンプニング情報が削除されます。

（注） ルート ダンプニングがイネーブルの場合は、リセットによってルートが取り消されるときでも、BGP ピアのリセットにペナルティは適用されません。

BGP ルート ダンプニングの設定

BGP ルート ダンプニングを設定してモニタするには、次の作業を実行します。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 4 **bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*]

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)# bgp dampening 30 1500 10000 120
```

指定したアドレス ファミリに対して BGP ダンプニングを設定します。

ステップ 5 **commit**

ルーティング ポリシーの強制適用

外部 BGP（eBGP）ネイバーには、インバウンドおよびアウトバウンドのポリシーを設定する必要があります。ポリシーが設定されていない場合、そのネイバーからのルートは受け入れられず、いずれのルートもそのネイバーにアドバタイズされません。この付加的なセキュリティ手段によって、設定を誤って省略した場合に、ルートが偶然受け入れられたり、アドバタイズされたりすることが決してなくなります。

（注） この制約は eBGP ネイバー（このルータと異なる自律システムに属すネイバー）だけに適用されます。内部 BGP（iBGP）ネイバー（同じ自律システム内のネイバー）の場合は、ポリシーがなければ、すべてのルートが受け入れられるか、アドバタイズされます。

ルーティング テーブル更新時のポリシーの適用

BGPのテーブルポリシー機能を使用すると、ルートのトラフィック索引の値を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に設定できます。この機能をイネーブルにするにはtable-policyコマンドを使用します。またBGPポリシー アカウンティング機能もサポートされています。テーブルポリシーを使用すると、一致基準に基づいてRIBからのルートをドロップすることもできます。この機能は特定のアプリケーションにおいて有用ですが、BGP がグローバル ルーティングおよびフォワーディング テーブルにインストールしていないネイバーに対して、BGP がルートをアドバタイズするところに、簡単にルーティング「ブラック ホール」が作成されてしまうため、注意して使用する必要があります。

ルーティングテーブルにインストールされるルートにルーティングポリシーを適用するには、次の作業を実行します。

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 120.6
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

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

例：

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

IPv4 またはIPv6のいずれかのアドレス ファミリを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 4 **table-policy** *policy-name*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)# table-policy tbl-plcy-A
```

ルーティング テーブルにインストールされるルートに、指定されたポリシーを適用します。

ステップ 5 **commit**

ルーティング ポリシーの適用：例

次の例では、すべてのルートが変更なしで許可およびアドバタイズされる場合に、eBGP ネイバーに対して単純な pass-all ポリシーが設定されています。

```
RP/0/RP0/cpu 0: router(config)# route-policy pass-all
RP/0/RP0/cpu 0: router(config-rpl)# pass
RP/0/RP0/cpu 0: router(config-rpl)# end-policy
RP/0/RP0/cpu 0: router(config)# commit
```

ネイバーに pass-all ポリシーを適用するには、ネイバー アドレス ファミリ コンフィギュレーション モードで **route-policy (BGP)** コマンドを使用します。次の例は、ネイバー 192.168.40.42 からの受信と、このネイバーに対するすべての IPv4 ユニキャスト ルートのアドバタイズを、すべての IPv4 ユニキャスト ルートに許可する方法を示します。

```
RP/0/RP0/cpu 0: router(config)# router bgp 1
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 192.168.40.24
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 21
RP/0/RP0/cpu 0: router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# route-policy pass-all in
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# route-policy pass-all out
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# commit
```

すべてのアクティブ アドレス ファミリに対するインバウンドとアウトバウンドの両方のポリシーを持っていない eBGP ネイバーを表示するには、**show bgp summary** コマンドを使用します。次の例の出力では、該当する eBGP ネイバーが感嘆符（!）によって示されています。

```
RP/0/RP0/cpu 0: router# show bgp all all summary

Address Family: IPv4 Unicast
============================

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 41
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process         RecvTblVer    bRIB/RIB  SendTblVer
Speaker                41          41          41

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  St/PfxRcd
10.0.101.1        0     1     919     925       41    0    0 15:15:08       10
10.0.101.2        0     2       0       0        0    0    0 00:00:00 Idle
```

BGP ネイバー グループおよびネイバーの設定

BGP ネイバー グループを設定し、ネイバーにネイバー グループの設定を適用するには、次の作業を実行します。ネイバー グループは、ネイバーに関連するアドレス ファミリから独立した設定とアドレス ファミリ固有の設定を持つテンプレートです。

ネイバー グループを設定すると、各ネイバーは、**use** コマンド経由で設定を継承できるようになります。ネイバー グループを使用するように設定されているネイバーは、デフォルトでネイバー グループの設定すべて（アドレス ファミリに依存しない設定とアドレス ファミリ固有の設定を含む）を継承します。継承された設定を上書きするには、ネイバーに対して直接コマンドを設定するか、または **use** コマンドを使用して、セッション グループ、またはアドレス ファミリ グループを設定します。

ネイバー グループではアドレス ファミリに依存しない設定を行うことができます。アドレス ファミリ固有の設定では、アドレス ファミリ サブモードを開始するようにネイバー グループのアドレス ファミリを設定する必要があります。ネイバー グループ コンフィギュレーション モードでは、ネイバー グループについて、アドレス ファミリに依存しないパラメータを設定できます。ネイバー グループ コンフィギュレーション モードで **address-family** コマンドを使用します。**neighbor group** コマンドを使用してネイバー グループ名を指定した後で、オプションをそのネイバー グループに割り当てることができます。

（注） 指定されたネイバー グループで設定できるコマンドはすべて、ネイバーでも設定できます。

（注） 6.3.2 よりも前の Cisco IOS-XR のバージョンでは、BGP ネイバーに属している自律システムを削除したり、単一の IOS-XR commit を使用して BGP ネイバーグループに移動することはできません。6.3.2 以降では、自律システムをネイバーから単一の IOS-XR commit 内のネイバー グループに移動できます。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 4 exit

例：

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

現在のコンフィギュレーション モードを終了します。

ステップ 5 **neighbor-group** *name*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor-group nbr-grp-A
```

ルータをネイバー グループ コンフィギュレーション モードにします。

ステップ 6 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbrgrp)# remote-as 2002
```

ネイバーを作成し、リモート自律システム番号を割り当てます。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 8 **route-policy** *route-policy-name* { **in** | **out** }

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbrgrp-af)# route-policy drop-as-1234 in
```

（任意）指定したポリシーを着信 IPv4 ユニキャスト ルートに適用します。

ステップ 9 **exit**

例：

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

現在のコンフィギュレーション モードを終了します。

ステップ 10 **exit**

例：

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

現在のコンフィギュレーション モードを終了します。

ステップ 11 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.168.40.24
```

BGPルーティングのためにルータをネイバーコンフィギュレーションモードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

ステップ 12 **use neighbor-group** *group-name*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# use neighbor-group nbr-grp-A
```

（任意）BGPネイバーが指定されたネイバーグループから設定を継承することを指定します。

ステップ 13 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 2002
```

ネイバーを作成し、リモート自律システム番号を割り当てます。

ステップ 14 **commit**

BGP ネイバー設定：例

情報を共有するように自律システムのBGPネイバーを設定する例を次に示します。この例では BGP ルータを自律システム 109 に割り当て、自律システムの送信元として 2 つのネットワークのリストが表示される例を示します。3 つのリモート ルータ（とその自律システム）のアドレスのリストが表示されます。設定したルータは、ネットワーク 172 .16 .0.0 および 192.168 .7.0 に関する情報を隣接ルータと共有します。リストの 1 番目のルータは別の自律システムにあり、2 番目の **neighbor** および **remote-as** コマンドによってアドレス 172 .26 .234.2 の内部ネイバーが（同じ自律システム番号を使用して）指定され、3 番目の **neighbor** および **remote-as** コマンドによって別の自律システムのネイバーが指定されます。

```
route-policy pass-all
 pass
end-policy
router bgp 109
 address-family ipv4 unicast
  network 172.16.0.0 255.255.0.0
  network 192.16831.7.0 255.255.0.0
  neighbor 172.16.200.1
   remote-as 167
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-out out
  neighbor 172.26.234.2
   remote-as 109
   exit
 address-family ipv4 unicast
  neighbor 172.26.64.19
   remote-as 99
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
```

BGP ネイバーの無効化

設定を削除せずにネイバーを管理シャットダウンするには、次の作業を実行します。

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 127
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.168.40.24
```

BGPルーティングのためにルータをネイバー コンフィギュレーション モードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

ステップ 4 **shutdown**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# shutdown
```

指定されたネイバーのすべてのアクティブ セッションをディセーブルにします。

ステップ 5 **commit**

BGP インバウンド ソフト リセットを使用したネイバーのリセット

指定されたグループまたはネイバーの指定アドレス ファミリに対してインバウンド ソフト リセットをトリガーするには、次の作業を実行ます。グループは、 ***** 、 *ip-address* 、 *as-number* 、または **external** キーワードおよび引数によって指定されます。

ネイバーのインバウンドポリシーまたはアウトバウンドポリシーを変更する場合、またはルーティングアップデートの送信または受信に影響を与えるその他の設定を変更する場合には、ネイバーのリセットが便利です。インバウンドソフトリセットがトリガーされた場合、ネイバーが ROUTE_REFRESH 機能をアドバタイズしていれば、BGP はデフォルトでこのネイバーに REFRESH 要求を送信します。ネイバーが ROUTE_REFRESH 機能をアドバタイズしているかどうかを判別するには、**show bgp neighbors** コマンドを使用します。

手順

	コマンドまたはアクション	目的
ステップ 1	**show bgp neighbors** **例**： `RP/0/RP0/cpu 0: router# show bgp neighbors`	ネイバーから受信したルート リフレッシュ機能がイネーブルであることを確認します。
ステップ 2	**soft** [**in** [**prefix-filter**] | **out**] **例**： `RP/0/RP0/cpu 0: router# clear bgp ipv4 unicast 10.0.0.1 soft in`	BGP ネイバーをソフト リセットします。 • ***** キーワードを指定すると、すべての BGP ネイバーがリセットされます。 • *ip-address* 引数では、リセットするネイバーのアドレスを指定します。 • *as-number* 引数では、自律システム番号に一致するすべてのネイバーがリセットされることを指定します。 • **external** キーワードは、すべての外部ネイバーがリセットされることを指定します。

BGP アウトバウンド ソフト リセットを使用したネイバーのリセット

指定されたグループまたはネイバーの指定アドレス ファミリに対してアウトバウンド ソフト リセットをトリガーするには、次の作業を実行します。グループは、*****、*ip-address*、*as-number*、または **external** キーワードおよび引数によって指定されます。

ネイバーのアウトバウンド ポリシーまたはアウトバウンド ポリシーを変更する場合、またはルーティング アップデートの送信または受信に影響を与えるその他の設定を変更する場合には、ネイバーのリセットが便利です。

アウトバウンド ソフト リセットがトリガーされると、BGP は、このアドレス ファミリに対するルートをすべて、指定されたネイバーに再送信します。

ネイバーが ROUTE_REFRESH 機能をアドバタイズしているかどうかを判別するには、**show bgp neighbors** コマンドを使用します。

手順

	コマンドまたはアクション	目的
ステップ 1	**show bgp neighbors** 例： `RP/0/RP0/cpu 0: router# show bgp neighbors`	ネイバーから受信したルート リフレッシュ機能がイネーブルであることを確認します。
ステップ 2	例： `RP/0/RP0/cpu 0: router# clear bgp ipv4 unicast 10.0.0.2 soft out`	BGP ネイバーをソフト リセットします。 • ***** キーワードを指定すると、すべての BGP ネイバーがリセットされます。 • *ip-address* 引数では、リセットするネイバーのアドレスを指定します。 • *as-number* 引数では、自律システム番号に一致するすべてのネイバーがリセットされることを指定します。 • **external** キーワードは、すべての外部ネイバーがリセットされることを指定します。

BGP ハード リセットを使用したネイバーのリセット

ハード リセットを使用してネイバーをリセットするには、次の作業を実行します。ハード リセットにより、ネイバーへの TCP 接続が削除され、ネイバーから受信したすべてのルートが BGP テーブルから削除され、その後このネイバーとのセッションが再確立されます。 **graceful**

キーワードを指定すると、ネイバーからのルートは BGP テーブルから即座に削除されず、古い（stale）ルートとしてマークされます。セッションの再確立後、ネイバーから再受信されなかった古いルートはすべて削除されます。

手順

clear bgp { **ipv4** { **unicast** | **labeled-unicast** | **all** | **tunnel tunnel** | **mdt** } | **ipv6 unicast** | **all** | **labeled-unicast** } | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** | **mdt** | **tunnel** } | **vpnv4 unicast** | **vrf** { *vrf-name* | **all** } { **ipv4 unicast** | **labeled-unicast** } | **ipv6 unicast** } | **vpnv6 unicast** } { ***** | *ip-address* | **as** *as-number* | **external** } [**graceful**] **soft** [**in** [**prefix-filter**] | **out**] **clear bgp** { **ipv4 | ipv6**} { **unicast** | **labeled-unicast** }

例：

```
RP/0/RP0/cpu 0: router# clear bgp ipv4 unicast 10.0.0.3
```

BGP ネイバーをクリアします。

- ***** キーワードを指定すると、すべての BGP ネイバーがリセットされます。
- *ip-address* 引数では、リセットするネイバーのアドレスを指定します。
- *as-number* 引数では、自律システム番号に一致するすべてのネイバーがリセットされることを指定します。
- **external** キーワードは、すべての外部ネイバーがリセットされることを指定します。

graceful キーワードはグレースフル リスタートを指定します。

ネイバーからのソフトウェアツーストア更新の設定

ネイバーからソフトウェアツーストア更新を受信するように設定するには、次の作業を実行します。

ネイバーがルート リフレッシュに対応している場合は、soft-reconfiguration inbound コマンドによって、ルート リフレッシュ要求がネイバーに送信されるようになります。ネイバーがルート リフレッシュに対応していない場合は、ネイバーが受信ルートを再学習するようにするため、**clear bgp soft** コマンドを使用してネイバーをリセットする必要があります。

（注） ネイバーからのアップデートの保存は、ネイバーがルート リフレッシュに対応しているか、soft-reconfiguration inbound コマンドが設定されている場合にだけ機能します。ネイバーがルート リフレッシュに対応しており、**soft-reconfiguration inbound** コマンドが設定されていても、このコマンドで **always** オプションが使用されていない場合は元のルートは格納されません。元のルートはルート リフレッシュ要求によって容易に復元できます。ルート リフレッシュは、ルーティング情報を再送信するためにピアに要求を送信します。soft-reconfiguration inbound コマンドは、変更されていない形式でピアから受信したすべてのパスを保存し、クリアする際にこれらの保存されたパスを参照します。ソフト再設定はメモリに負荷がかかる処理です。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.168.40.24
```

BGPルーティングのためにルータをネイバー コンフィギュレーションモードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 5 **soft-reconfiguration inbound** [**always]**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# soft-reconfiguration inbound always
```

指定したネイバーから受信したアップデートを格納するようにソフトウェアを設定します。ソフト再設定インバウンドを設定すると、ソフトウェアは変更またはフィルタ処理されたルートのほかに、元の変更されていないルートを格納することになります。これにより、インバウンド ポリシーの変更後に「ソフト クリア」を実行できるようになります。

ソフト再設定により、ピアがルート フレッシュに対応していない場合、ソフトウェアはポリシー適用前に受信した更新を格納できます（対応している場合は更新のコピーが格納されます）。 **always** キーワードを使用すると、ルート リフレッシュがピアでサポートされている場合でも、ソフトウェアにコピーが格納されます。

ステップ 6 **commit**

ネイバーの変更の記録

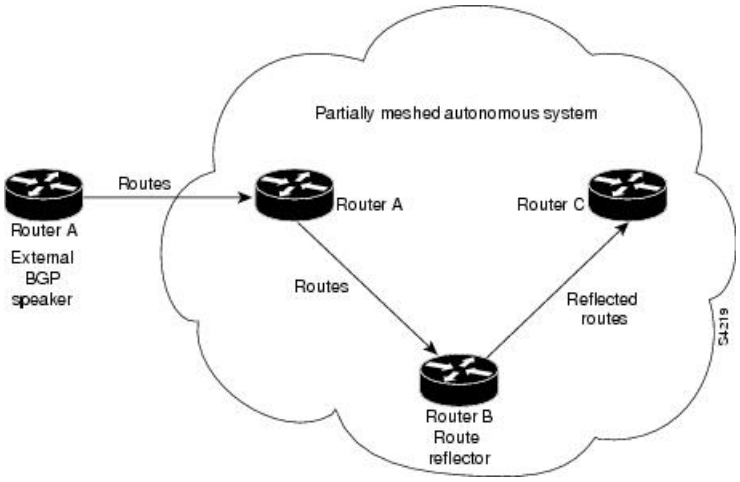
ネイバー変更のロギングはデフォルトでイネーブルになっています。ロギングをオフにするには、**log neighbor changes disable** コマンドを使用します。ロギングがディセーブルにされている場合にロギングを再びイネーブルにするには、**no log neighbor changes disable** コマンドを使用します。

BGP ルート リフレクタ

BGP を使用するには、すべての iBGP スピーカーが完全メッシュ化されている必要があります。ただし、iBGP スピーカーの数が多い場合、この要件には適切な拡張性がありません。コンフェデレーションを設定する代わりに、ルート リフレクタ設定を使用すると iBGP メッシュを削減できます。ルート リフレクタがある場合は、学習したルートをネイバーに渡す方法があるため、すべての iBGP スピーカーを完全にメッシュ化する必要はありません。このモデルでは、iBGP が学習したルートを一連の iBGP ネイバーに渡す役割を持つルート リフレクタとして、1 つの iBGP ピアを設定しています。

図 2 : ルート リフレクタのある単純な BGP モデル （52 ページ） では、ルータ B がルート リフレクタとして設定されています。ルータ A からアドバタイズされたルートをルート リフレクタが受信すると、ルータ C にアドバタイズします。逆の場合も同じです。このスキームにより、ルータ A とルータ C 間の iBGP セッションは不要になります。

図 *2:* ルート リフレクタのある単純な *BGP* モデル



ルータ リフレクタの詳細については、BGP ルート リフレクタ リファレンス （147 ページ）を参照してください。

BGP のルート リフレクタの設定

BGP のルート リフレクタを設定するには、次の作業を実行します。

route-reflector-client コマンドで設定されるネイバーはすべてクライアント グループのメンバーであり、その他の iBGP ピアはローカル ルータ リフレクタの非クライアント グループのメンバーです。

ルートリフレクタは、そのクライアントとあわせてクラスタを形成します。クライアントからなるクラスタには通常、ルートリフレクタが 1 つ存在します。このようなインスタンスでは、クラスタはソフトウェアにより、ルート リフレクタのルータ ID と認識されます。冗長性を高め、ネットワークでのシングルポイント障害を回避するために、クラスタに複数のリフレクタが含まれていることもあります。この場合、このクラスタのルートリフレクタはすべて、同じ 4 バイトのクラスタ ID を使って設定する必要があります。これはルート リフレクタが、同じクラスタに属する別のルート リフレクタからのアップデートを認識できるようにするためです。クラスタに複数のルータ リフレクタがある場合にクラスタ ID を設定するには、**bgp cluster-id** コマンドを使用します。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **bgp cluster-id** *cluster-id*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# bgp cluster-id 192.168.70.1
```

クラスタに対応するルート リフレクタの 1 つとして、ローカル ルータを設定します。クラスタを識別するために、指定したクラスタ ID を設定します。

ステップ 4 **neighbor** *ip-address*

例：

```
  RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.168.40.24
```

BGP ルーティングのためにルータをネイバー コンフィギュレーション モードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

ステップ 5 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 2003
```

ネイバーを作成し、リモート自律システム番号を割り当てます。

ステップ 6 **address-family** { **ipv4** | **ipv6** } **unicast**

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 7 **route-reflector-client**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# route-reflector-client
```

BGP ルート リフレクタとしてルータを設定し、そのクライアントとしてネイバーを設定します。

ステップ 8 **commit**

BGP ルート リフレクタ：例

次に、アドレス ファミリを使用して、内部 BGP ピア 10.1.1.1 をユニキャスト プレフィックスのリフレクタ クライアントとして設定する例を示します。

```
router bgp 140
 address-family ipv4 unicast
  neighbor 10.1.1.1
   remote-as 140
   address-family ipv4 unicast
    route-reflector-client
    exit
```

ルート ポリシーによる BGP ルート フィルタリングの設定

ルート ポリシーによる BGP ルーティング フィルタリングを設定するには、次の作業を実行します。

手順

	コマンドまたはアクション	目的
ステップ 1	**configure**	

ステップ 2	**route-policy** *name*
	例：
	`RP/0/RP0/cpu 0: router(config)# route-policy drop-as-1234`
	`RP/0/RP0/cpu 0: router(config-rpl)# if as-path passes-through '1234' then`
	`RP/0/RP0/cpu 0: router(config-rpl)# apply check-communities`
	`RP/0/RP0/cpu 0: router(config-rpl)# else`
	`RP/0/RP0/cpu 0: router(config-rpl)# pass`
	`RP/0/RP0/cpu 0: router(config-rpl)# endif`
		（任意）ルートポリシーを作成し、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。このモードではルート ポリシーを定義できます。

ステップ 3	**end-policy** 例： `RP/0/RP0/cpu 0: router(config-rpl)# end-policy`	（任意）ルート ポリシーの定義を終了し、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。

	コマンドまたはアクション	目的
ステップ 4	**router bgp** *as-number* 例： `RP/0/RP0/cpu 0: router(config)# router bgp 120`	自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。
ステップ 5	**neighbor** *ip-address* 例： `RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.168.40.24`	BGP ルーティングのためにルータをネイバー コンフィギュレーション モードにして、ネイバーの IP アドレスを BGP ピアとして設定します。
ステップ 6	**address-family** { **ipv4** | **ipv6** } **unicast** 例： `RP/0/RP0/cpu 0: router(config-bgp-nbr)# address-family ipv4 unicast`	IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。 このコマンドのすべてのキーワードと引数のリストを参照するには、CLIヘルプ（?）を使用します。
ステップ 7	**route-policy** *route-policy-name* { **in** | **out** } 例： `RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# route-policy drop-as-1234 in`	指定されたポリシーをインバウンドルートに適用します。
ステップ 8	**commit**	

BGP 属性フィルタリングの設定

BGP 属性フィルタは、BGP アップデート メッセージ内の BGP アップデートの整合性を確認し、無効な属性を検出したときは応答を最適化します。BGP アップデート メッセージには、必須およびオプションの属性のリストが含まれています。アップデートメッセージ内のこれらの属性には、MED、LOCAL_PREF、COMMUNITY などがあります。場合によって、属性が不正である場合は、ルータの受信側でこれらの属性をフィルタリングする必要があります。BGP 属性フィルタ機能では、着信アップデート メッセージで受信した属性をフィルタリングします。属性フィルタは、受信側ルータで好ましくない動作を引き起こす可能性のある属性を排除するためにも使用できます。BGP アップデートの中には、ネットワーク層到達可能性情報（NLRI）またはアップデート メッセージ内の他のフィールドなどの誤った形式の属性のために、形式が不正になるものがあります。これらの不正なアップデートを受信すると、受信側ルータで好ましくない動作が発生します。このような不正な動作は、アップデートメッセージ

の解析時や、受信した NLRI の再アドバタイズ時に発生することがあります。このような場合に備えて、受信側でこれらの破損した属性をフィルタ処理することが重要です。

属性フィルタリングを設定するには、1 つまたはある範囲の属性コードと対応するアクションを指定します。受信したアップデートメッセージに1つ以上のフィルタされた属性が含まれている場合、メッセージに対して設定されたアクションが実行されます。オプションで、さらに詳細なデバッグを行うためにアップデート メッセージを保存して、コンソールに syslog メッセージを表示することもできます。属性がフィルタと一致した場合は、属性のその後の処理は停止され、対応するアクションが実行されます。BGP 属性フィルタリングを設定するには、次のタスクを実行します。

手順

ステップ 1 **configure**

ステップ 2 **router bgp** *as-number*

例：

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

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **attribute-filter group** *attribute-filter group name*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# attribute-filter group ag_discard_med
```

属性フィルタ グループ名を指定し、属性フィルタ グループ コンフィギュレーション モードを開始することで、BGP ネイバーに特定の属性フィルタ グループを設定できます。

ステップ 4 **attribute** *attribute code* { **discard** | **treat-as-withdraw** }

例：

```
RP/0/RP0/cpu 0: router(config-bgp-attrfg)# attribute 24 discard
```

単一またはある範囲の属性コードと関連するアクションを指定します。実行できるアクションには次のものがあります。

- Treat-as-withdraw：アップデート メッセージを取り消すかを検討します。対応する IPv4 ユニキャストまたは MP_REACH NLRI があれば、ネイバーの Adj-RIB-In から取り消します。
- Discard Attribute：この属性を廃棄します。一致した部分の属性は廃棄され、アップデート メッセージの残りの部分は正常に処理されます。

BGP ネクスト ホップ トラッキング

ネクストホップ情報が変更されると、BGPはルーティング情報ベース（RIB）から通知を受信します（イベント駆動型の通知）。BGP は RIB からネクスト ホップ情報を取得して次の処理を行います。

- ネクスト ホップが到達可能であるかどうかを確認する。
- ネクスト ホップへの完全再帰 IGP メトリックを見つける（最適パス計算で使用）。
- 受信したネクスト ホップを検証する。
- 発信ネクスト ホップを計算する。
- ネイバーの到達可能性および接続を確認する。

BGP ネクスト ホップの参照 （142 ページ） で、BGP ネクスト ホップに関する追加の概念的な詳細を提供します。

BGP ネクスト ホップ トリガー遅延の設定

BGP ネクスト ホップ トリガー遅延を設定するには、次の作業を実行します。ルーティング情報ベース（RIB）では変更の重大度に基づいてダンプニング通知が分類されます。イベント通知はクリティカルおよび非クリティカルとして分類されます。この作業では、クリティカルイベントと非クリティカル イベントの最小バッチ間隔を指定できます。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 4 **nexthop trigger-delay** { **critical** *delay* | **non-critical** *delay* }

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)# nexthop trigger-delay critical 15000
```

重要なネクスト ホップ トリガー遅延を設定します。

ステップ 5 **commit**

BGP 更新でのネクスト ホップ処理のディセーブル化

ネイバーに対するネクスト ホップの計算をディセーブルにし、BGP アップデートのネクスト ホップ フィールドにユーザ自身のアドレスを挿入するには、次の作業を実行します。ルートをアドバタイズするときに使用する最適なネクストホップの計算をディセーブルにすると、すべてのルートがネットワーク デバイスによってネクスト ホップとしてアドバタイズされます。

（注） ネクストホップ処理は、アドレス ファミリ グループ、ネイバー グループ、またはネイバー アドレス ファミリに対して無効にすることができます。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.168.40.24
```

BGPルーティングのためにルータをネイバーコンフィギュレーションモードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

ステップ 4 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 206
```

ネイバーを作成し、リモート自律システム番号を割り当てます。

ステップ 5 **address-family { ipv4 | ipv6 } unicast**

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 6 **next-hop-self**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# next-hop-self
```

指定されたネイバーにアドバタイズされるすべてのルートのネクスト ホップ属性をローカル ルータのアドレスに設定します。ルートをアドバタイズするときに使用する最適なネクスト ホップの計算をディセーブルにすると、すべてのルートがローカル ネットワーク デバイスによってネクスト ホップとしてアドバタイズされます。

ステップ 7 **commit**

BGP コスト コミュニティ

BGP コスト コミュニティは非過渡的な拡張コミュニティ属性で、内部 BGP（iBGP）およびコンフェデレーション ピアへ渡されますが、外部 BGP（eBGP）ピアへは渡されません。コスト コミュニティ機能により、コスト値を特定のルートに割り当てることで、ローカル ルート プリファレンスをカスタマイズし、最適パス選択プロセスに反映させることができます。拡張コミュニティ形式は、最適パス アルゴリズムの異なるポイントでの最適パスの決定に影響する標準の挿入ポイント（POI）を定義します。

BGP コスト コミュニティの参照 （142 ページ） で、BGP コスト コミュニティに関する追加の概念的な詳細を提供します。

BGP コスト コミュニティの設定

BGP は同一宛先への複数のパスを受信し、最適パス アルゴリズムを使用して RIB にインストールする最適なパスを決定します。ユーザが部分比較後に出力点を決定できるようにするため、最適パス選択処理で同等パスのタイブレークのためにコスト コミュニティが定義されます。BGP コスト コミュニティを設定するには、次の作業を実行します。

手順

ステップ 1 **configure**

ステップ 2 **route-policy** *name*

例：

```
RP/0/RP0/cpu 0: router(config)# route-policy costA
```

ルート ポリシー コンフィギュレーション モードに切り替え、設定するルート ポリシーの名前を指定します。

ステップ 3 **set extcommunity cost** { *cost-extcommunity-set-name* | *cost-inline-extcommunity-set* } [**additive**]

例：

```
RP/0/RP0/cpu 0: router(config)# set extcommunity cost cost_A
```

コストの BGP 拡張コミュニティ属性を指定します。

ステップ 4 **end-policy**

例：

```
RP/0/RP0/cpu 0: router(config)# 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 5 **router bgp** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config)# router bgp 120
```

BGP コンフィギュレーション モードを開始します。このモードでは BGP ルーティング プロセスを設定できます。

ステップ 6 次のいずれかを実行します。

- **default-information originate**
- **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]
- **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute isis** *process-id* [**level** { **1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]

コスト コミュニティを付加ポイント（ルート ポリシー）に適用します。

ステップ 7 次のいずれかを実行します。

- **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **network** { *ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*]
- **neighbor** *ip-address* **remote-as** *as-number*
- **route-policy** *route-policy-name* { **in** | **out** }

ステップ 8 **commit**

ステップ 9 **show bgp** *ip-address*

例：

```
RP/0/RP0/cpu 0: router# show bgp 172.168.40.24
```

コスト コミュニティを次の形式で表示します。

Cost: *POI* : *cost-community-ID* : *cost-number*

BGPコミュニティおよび拡張コミュニティアドバタイズメントの設定

コミュニティ属性および拡張コミュニティ属性を eBGP ネイバーに送信することを指定するには、次の作業を実行します。これらの属性は、デフォルトでは eBGP ネイバーに送信されません。これに対して、iBGP ネイバーには常に送信されます。ここでは、コミュニティ属性を送信できるようにする方法の例を示します。拡張コミュニティを送信できるようにするには、**send-community-ebgp** キーワードを **send-extended-community-ebgp** キーワードで置き換えます。

send-community-ebgp コマンドをネイバー グループまたはアドレス ファミリ グループに対して設定すると、このグループを使用するすべてのネイバーが設定を継承します。あるネイバーに対して特別にこのコマンドを設定すると、継承された値が上書きされます。

（注） BGP コミュニティと拡張コミュニティ フィルタリングは、iBGP ネイバーには設定できません。コミュニティと拡張コミュニティは、VPNv4、MDT、IPv4、および IPv6 アドレス ファミリでは常に iBGP ネイバーに送信されます。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.168.40.24
```

BGPルーティングのためにルータをネイバーコンフィギュレーションモードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

ステップ 4 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 2002
```

ネイバーを作成し、リモート自律システム番号を割り当てます。

ステップ 5 **address-family**{**ipv4** {**labeled-unicast** | **unicast** | **mdt** | | **mvpn** | **rt-filter** | **tunnel**} | **ipv6** {**labeled-unicast** | **mvpn** | **unicast**}}

例：

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

指定のアドレス ファミリに対応しネイバーアドレス ファミリ コンフィギュレーション モードを開始します。**ipv4** または **ipv6** アドレス ファミリ キーワードと、指定したアドレス ファミリ サブ モード ID の 1 つを使用します。

IPv6 アドレス ファミリ モードでは、次のサブモードをサポートしています。

- **labeled-unicast**
- **mvpn**
- **unicast**

IPv4 アドレス ファミリ モードでは、次のサブモードをサポートしています。

- **labeled-unicast**
- **mdt**
- **mvpn**
- **rt-filter**
- **tunnel**
- **unicast**

ステップ 6 次のいずれかのコマンドを使用します。

- **send-community-ebgp**
- **send-extended-community-ebgp**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# send-community-ebgp
```

または

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# send-extended-community-ebgp
```

ルータが（デフォルトでは eBGP ネイバーでディセーブルにされている）コミュニティ属性と拡張コミュニティ属性を指定された eBGP ネイバーに送信することを指定します。

ステップ 7 **commit**

BGP の大型コミュニティの設定

BGP コミュニティはコミュニティ属性を使用して、宛先をグループ化し、宛先グループでの承認、拒否、優先、または再配布などのルーティングの決定を適用する方法を提供します。BGP コミュニティ属性は、2 つの 16 ビット部分に分割される 1 つ以上の 4 バイト値で構成される可変長属性です。上位の 16 ビットが AS 番号を表し、下位ビットが AS の演算子によって割り当てられたローカルに定義された値を表します。

4 バイトの ASN（RFC6793）の採用以降、4 バイトの ASN、およびルートにタグ付けする AS 固有の値をエンコードするのに 4 バイトでは不十分なため、BGP コミュニティ属性は 4 バイトの ASN に対応できなくなりました。BGP 拡張コミュニティは、グローバル管理者フィールドとして 4 バイトの AS のエンコードを許可しますが、ローカル管理者フィールドには利用可能なスペースが 2 バイトしかありません。そのため、6 バイトの拡張コミュニティ属性も適切ではありません。この制限を打開するには、12 バイトの BGP 大型コミュニティを設定します。これはオプションの属性であり、自律システム番号をグローバル管理者としてエンコードする最上位 4 バイト値と、ローカル値をエンコードする残りの 4 バイトの割り当て済みの数字を提供します。

BGP コミュニティと同様に、ルータはルートポリシー言語（RPL）を使用して BGP 大型コミュニティを BGP ルータに適用でき、他のルータはルートに付加されたコミュニティに基づいてアクションを実行できます。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。

他のコマンドで大型コミュニティを指定する場合は、コロンで区切った 3 つの負ではない 10 進整数として指定します（たとえば、1:2:3）。各整数は 32 ビットで格納されます。各整数の有効な範囲は 0 ～ 4294967295 です。

ルートポリシー ステートメントでは、BGP 大型コミュニティの各整数を次のいずれかの表現で置き換えることができます。

- [x..y]：この表現は、x と y の範囲（両端の値を含む）を指定します。
- *：この表現は任意の数値を表します。
- peeras：この表現は、必要に応じてコミュニティの送信元または送信先のネイバーの AS 番号で置き換えられます。
- not-peeras：この表現は、peeras 以外の任意の数値と一致します。

- private-as：この表現は、プライベート ASN 範囲（[64512..65534] および [4200000000..4294967294]）の任意の数値を指定します。

これらの表現は、ポリシー一致ステートメントでも使用できます。

IOS 正規表現（ios-regex）と DFA 形式の正規表現（dfa-regex）は、大型コミュニティ ポリシーのすべての match 文と delete 文に使用できます。たとえば、IOS 正規表現 ios-regex '^5:.*:7$' は、表現 5:*:7 と同等です。

send-community-ebgp コマンドは、BGP 大型コミュニティを含むように拡張されています。BGP スピーカーで大型コミュニティを ebgp ネイバーに送信するには、このコマンドが必要です。

制限とガイドライン

次に、BGP 大型コミュニティに適用される制限とガイドラインを示します。

- BGP コミュニティ属性のすべての機能を BGP 大型コミュニティ属性に使用できます。
- BGP スピーカーで大型コミュニティを ebgp ネイバーに送信するには、**send-community-ebgp** コマンドが必要です。
- よく知られた大型コミュニティはありません。
- peeras 表現は、大型コミュニティ セットでは使用できません。
- peeras 表現は、neighbor-in または neighbor-out 付加ポイントで適用されるルート ポリシーに含まれる、大型コミュニティの match 文または delete 文でのみ使用できます。
- not-peeras 表現は、大型コミュニティセットまたはポリシーの set 文では使用できません。

設定例：大型コミュニティ セット

大型のコミュニティ セットは 1 セットの大型コミュニティを定義します。ルートポリシーの match 文および set 文では、名前付きの大型コミュニティ セットが使用されます。

次の例は、名前付きの大型コミュニティ セットを作成する方法を示しています。

```
RP/0/RP0/CPU0:router(config)# large-community-set catbert
RP/0/RP0/CPU0:router(config-largecomm)#  1: 2: 3,
RP/0/RP0/CPU0:router(config-largecomm)#  peeras:2:3
RP/0/RP0/CPU0:router(config-largecomm)# end-set
```

設定例：大型コミュニティの設定

次の例に、**set large-community** {*large-community-set-name* | *inline-large-community-set* | *parameter* } [**additive**] コマンドを使用して、ルートで BGP 大型コミュニティ属性を設定する方法を示します。名前付きの大型コミュニティセットまたはインライン セットを指定できます。**additive** キーワードは、ルート内にすでに存在する大型コミュニティを保持し、新しい大型コミュニティのセットを追加します。ただし、**additive** キーワードを指定してもエントリが重複することはありません。

特定の大型コミュニティがルートに付加されている場合に、set 文の additive キーワードで同じ大型コミュニティを再度指定しても、指定した大型コミュニティは再追加されません。マージ操作を行うと、重複エントリが削除されます。これは、peeras キーワードにも適用されます。

この例の peeras 表現は、必要に応じて BGP 大型コミュニティの送信元または送信先のネイバーの AS 番号で置き換えられます。

```
RP/0/RP0/CPU0:router(config)# route-policy mordac
RP/0/RP0/CPU0:router(config-rpl)# set large-community (1:2:3, peeras:2:3)
RP/0/RP0/CPU0:router(config-rpl)# end-set
RP/0/RP0/CPU0:router(config)# large-community-set catbert
RP/0/RP0/CPU0:router(config-largecomm)#  1: 2: 3,
RP/0/RP0/CPU0:router(config-largecomm)#  peeras:2:3
RP/0/RP0/CPU0:router(config-largecomm)# end-set
RP/0/RP0/CPU0:router(config)# route-policy wally
RP/0/RP0/CPU0:router(config-rpl)# set large-community catbert additive
RP/0/RP0/CPU0:router(config-rpl)# end-set
```

この例では、ASN が 1 のネイバーにルートポリシー mordac が適用されると、大型コミュニティ (1:2:3) が一度だけ設定されます。

（注） 大型コミュニティを ebgp ネイバーに送信するには、**send-community-ebgp** コマンドを設定する必要があります。

設定例：大型コミュニティの matches-any

次の例に、大型コミュニティ セットの要素で一致を確認するルート ポリシーの設定方法を示します。これはブール型の条件であり、ルート内の大型コミュニティのいずれかが、一致条件内の大型コミュニティのいずれかに一致した場合に true を返します。

```
RP/0/RP0/CPU0:router(config)# route-policy elbonia
RP/0/RP0/CPU0:router(config-rpl)# if large-community matches-any (1:2:3, 4:5:*) then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 94
RP/0/RP0/CPU0:router(config-rpl)#  endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

設定例：大型コミュニティの matches-every

次の例は、ステートメント内のすべての match 指定がルート内の 1 つ以上の大型コミュニティに一致する必要があるルート ポリシーの設定方法を示しています。

```
RP/0/RP0/CPU0:router(config)# route-policy bob
RP/0/RP0/CPU0:router(config-rpl)# if large-community matches-every (*:*:3, 4:5:*) then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 94
RP/0/RP0/CPU0:router(config-rpl)#  endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

この例では、次の大型コミュニティ セットを含むルートが TRUE を返します。

- (1:1:3, 4:5:10)
- (4:5:3)：この単一の大型コミュニティは両方の仕様に一致します。
- (1:1:3, 4:5:10, 7:6:5)

次の大型コミュニティ セットを含むルートは FALSE を返します。

(1:1:3, 5:5:10)：指定 (4:5:*) は一致しません。

設定例：大型コミュニティの matches-within

次の例に、大型コミュニティ セット内で照合するルート ポリシーの設定方法を示します。これは **large-community matches-any** コマンドに似ていますが、ルート内のすべての大型コミュニティが 1 つ以上の match 指定に一致する必要があります。大型コミュニティがないルートは一致することに注意してください。

```
RP/0/RP0/CPU0:router(config)# route-policy bob
RP/0/RP0/CPU0:router(config-rpl)# if large-community matches-within (*:*:3, 4:5:*) then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 103
RP/0/RP0/CPU0:router(config-rpl)#  endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

たとえば、次の大型コミュニティ セットを含むルートは TRUE を返します。

- (1:1:3, 4:5:10)
- (4:5:3)
- (1:2:3, 6:6:3, 9:4:3)

次の大型コミュニティ セットを含むルートは FALSE を返します。

(1:1:3, 4:5:10, 7:6:5)：大型コミュニティ (7:6:5) は一致しません

設定例：コミュニティの matches-within

次の例に、コミュニティ セットの要素内で照合するルート ポリシーの設定方法を示します。このコマンドは **community matches-any** コマンドに似ていますが、ルート内のすべてのコミュニティが 1 つ以上の match 指定に一致する必要があります。コミュニティがないルートは一致します。

```
RP/0/RP0/CPU0:router(config)# route-policy bob
RP/0/RP0/CPU0:router(config-rpl)# if community matches-within (*:3, 5:*)  then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 94
RP/0/RP0/CPU0:router(config-rpl)#  endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

たとえば、次のコミュニティ セットを含むルートは TRUE を返します。

- (1:3, 5:10)
- (5:3)
- (2:3, 6:3, 4:3)

次のコミュニティ セットを含むルートは FALSE を返します。

(1:3, 5:10, 6:5)：コミュニティ (6:5) は一致しません。

設定例：大型コミュニティの is-empty

次の例では、**large-community is-empty** 句を使用した、大型コミュニティ属性が設定されていないルートのフィルタリングを示します。

```
RP/0/RP0/CPU0:router(config)# route-policy lrg_comm_rp4
RP/0/RP0/CPU0:router(config-rpl)# if large-community is-empty then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 104
RP/0/RP0/CPU0:router(config-rpl)#  endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

設定例：属性フィルタ グループ

次の例に、大型コミュニティ属性を使用して属性フィルタ グループを設定し、BGP ネイバーに適用する方法を示します。フィルタは、BGP のパス属性と、BGP アップデート メッセージの受信時に実行するアクションを指定します。BGP ネイバーから指定の属性のいずれかが含まれているアップデートメッセージを受信すると、指定したアクションが実行されます。この例では、dogbert という属性フィルタが作成されて BGP ネイバー 10.0.1.101 に適用されます。このフィルタは、大型コミュニティ属性と破棄アクションを指定します。つまり、ネイバー 10.0.1.101 からの BGP アップデート メッセージで大型コミュニティの BGP パス属性を受信した場合、メッセージを処理する前にその属性が破棄されます。

```
RP/0/RP0/CPU0:router(config)# router bgp 100
RP/0/RP0/CPU0:router(config-bgp)# attribute-filter group dogbert
RP/0/RP0/CPU0:router(config-bgp-attrfg)# attribute LARGE-COMMUNITY discard
RP/0/RP0/CPU0:router(config-bgp-attrfg)# neighbor 10.0.1.101
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 6461
RP/0/RP0/CPU0:router(config-bgp-nbr)# update in filtering
RP/0/RP0/CPU0:router(config-nbr-upd-filter)# attribute-filter group dogbert
```

設定例：大型コミュニティの削除

次の例は、**delete large-community** コマンドを使用してルート ポリシーから指定の BGP 大型コミュニティを削除する方法を示しています。

```
RP/0/RP0/CPU0:router(config)# route-policy lrg_comm_rp2
RP/0/RP0/CPU0:router(config-rpl)# delete large-community in (ios-regex '^100000:')
RP/0/RP0/CPU0:router(config-rpl)# delete large-community all
RP/0/RP0/CPU0:router(config-rpl)# delete large-community not in (peeras:*:*, 41289:*:*)
```

確認

次の例では、**show bgp large-community***list-of-large-communities* [**exact-match**] コマンドで指定した大型コミュニティを含むルートが表示されます。オプション キーワード exact-match を使用すると、リストされるルートには指定した大型コミュニティのみが含まれます。このキーワードを指定しない場合は、表示されるルートに追加の大型コミュニティが含まれることがあります。

```
RP/0/0/CPU0:R1# show bgp large-community 1:2:3 5:6:7
Thu Mar 23 14:40:33.597 PDT
BGP router identifier 4.4.4.4, local AS number 3
BGP generic scan interval 60 secs
Non-stop routing is enabled
```

```
BGP table state: Active
Table ID: 0xe0000000   RD version: 66
BGP main routing table version 66
BGP NSR Initial initsync version 3 (Reached)
BGP NSR/ISSU Sync-Group versions 66/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
* 10.0.0.3/32         10.10.10.3               0     94      0 ?
* 10.0.0.5/32         10.11.11.5               0             0 5 ?
```

次の例では、**show bgp** *ip-address/ prefix-length* コマンドを使用して、ネットワークに接続されている大型コミュニティを表示します。

```
RP/0/0/CPU0:R4# show bgp 10.3.3.3/32
Thu Mar 23 14:36:15.301 PDT
BGP routing table entry for 10.3.3.3/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 42          42
Last Modified: Mar 22 20:04:46.000 for 18:31:30
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    10.11.11.5
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    10.11.11.5
  Local
    10.10.10.3 from 10.10.10.3 (10.3.3.3)
      Origin incomplete, metric 0, localpref 94, valid, internal, best, group-best
      Received Path ID 0, Local Path ID 0, version 42
      Community: 258:259 260:261 262:263 264:265
      Large Community: 1:2:3 5:6:7 4123456789:4123456780:4123456788
```

IGP への iBGP ルートの再配布

Intermediate System-to-Intermediate System（IS-IS）や Open Shortest Path First（OSPF）など、内部ゲートウェイ プロトコル（IGP）に iBGP ルートを再配布するには、次の作業を実行します。

（注） **bgp redistribute-internal** コマンドを使用するには、すべての BGP ルートを IP ルーティング テーブルに再インストールするために、**clear route *** コマンドを発行する必要があります。

注意 IGP への iBGP ルートの再配布は、自律システム内にルーティング ループが作成される原因となる可能性があります。このコマンドの使用には注意が必要です。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **bgp redistribute-internal**

例：

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

IGP（IS-IS や OSPF など）への iBGP ルートの再配布を許可します。

ステップ 4 **commit**

BGP への IGP の再配布

VRF アドレス ファミリへのプロトコルの再配布を設定するには、次の作業を実行します。

内部ゲートウェイ プロトコル（IGP）が PE-CE プロトコルとして使用されている場合でも、インポート ロジックは BGP を経由して実行されます。したがって、すべての IGP ルートを BGP VRF テーブルにインポートする必要があります。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **vrf** *vrf-name*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# vrf vrf_a
```

PE ルータで特定の VRF の BGP ルーティングをイネーブルにします。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 5 次のいずれかを実行します。

- **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute isis** *process-id* [**level** { **1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]

例：

```
RP/0/RP0/cpu 0: router(config-bgp-vrf-af)# redistribute ospf 1
```

VRF アドレス ファミリ コンテキストでプロトコルの再配布を設定します。

redistribute コマンドは、PE-CE ルータ間で BGP が使用されていない場合に使用します。PE-CE ルータ間で BGP が使用されている場合は、使用されている IGP を BGP に再配布して、他方の PE サイトとの VPN 接続を確立する必要があります。テーブル間でのインポートおよびエクスポートにも再配布が必要です。

ステップ 6 **commit**

アップデート グループ

BGP アップデート グループ機能には、アウトバウンド ポリシーを共有し、アップデート メッセージを共有できるネイバーのアップデート グループをダイナミックに計算し、最適化する新しいアルゴリズムが含まれています。BGP アップデート グループ機能では、アップデート グループ レプリケーションはピア グループ コンフィギュレーションから分離されるため、ネイバー コンフィギュレーションのコンバージェンス時間が短縮され、柔軟性が高まります。

BGP アップデート グループのモニタリング

この作業では、BGP アップデート グループの処理に関する情報を表示します。

手順

show bgp [**ipv4** { **unicast** | **multicast** | **all** | **tunnel** } | **ipv6** { **unicast** | **all** } | **all** { **unicast** | **multicast** | **all** **labeled-unicast** | **tunnel** } | **vpnv4 unicast** | **vrf** { *vrf-name* | **all** } [**ipv4** **unicast** **ipv6 unicast**] | **vpvn6 unicast**] **update-group** [**neighbor** *ip-address* | *process-id.index* [**summary** | **performance-statistics**]]

例：

```
RP/0/RP0/cpu 0: router# show bgp update-group 0.0
```

BGP アップデート グループの情報を表示します。

- *ip-address* 引数を指定すると、そのネイバーが属するアップデート グループが表示されます。
- *process-id.index* 引数では、表示する特定のアップデート グループを選択します。この引数は「プロセス ID（ドット）インデックス」の形式で指定します。プロセス ID の範囲は 0 ～ 254 です。インデックスの範囲は 0 ～ 4294967295 です。
- **summary** キーワードを指定すると、特定のアップデート グループに含まれているネイバーに関する要約情報が表示されます。
- このコマンドに引数を指定しないと、（指定したアドレス ファミリの）すべてのアップデート グループの情報が表示されます。
- **performance-statistics** キーワードを指定すると、アップデート グループのパフォーマンス統計情報が表示されます。

BGP アップデート グループの表示：例

次に、EXEC コンフィギュレーションXR EXEC モードで実行された **show bgp update-group** コマンドの出力例を示します。

```
show bgp update-group

 Update group for IPv4 Unicast, index 0.1:
   Attributes:
     Outbound Route map:rm
     Minimum advertisement interval:30
   Messages formatted:2, replicated:2
   Neighbors in this update group:
     10.0.101.92
```

```
Update group for IPv4 Unicast, index 0.2:
  Attributes:
    Minimum advertisement interval:30
  Messages formatted:2, replicated:2
  Neighbors in this update group:
    10.0.101.91
```

L3VPN iBGP PE-CE

L3VPN iBGP PE-CE 機能は、プロバイダー エッジ（PE）デバイスとカスタマー エッジ（CE）デバイス間でBGPルーティング情報を交換するiBGP（内部Border Gateway Protocol）セッションの確立に役立ちます。2つのBGP ピア間のBGP セッションは、それらのBGP ピアが同じ自律システム内に存在する場合に、iBGP セッションと呼ばれます。

L3VPN iBGP PE-CE の制限

次に、L3VPN iBGP PE-CE の設定に適用される制限を示します。

- IBGP PE CE 機能を切り替えてネイバーが route-refresh または soft-reconfiguration inbound をサポートしなくなった場合は、手動のセッション フラップを実行して変更を確認する必要があります。これが発生した場合は、次のメッセージが表示されます。

  ```
  RP/0/RP0/CPU0: %ROUTING-BGP-5-CFG_CHG_RESET: Internal VPN client configuration change
   on neighbor 10.10.10.1 requires HARD reset
  (clear bgp 10.10.10.1) to take effect.
  ```

- iBGP PE CE CLI 設定は、ネイバー/セッション グループを除き、デフォルト VRF のピアには使用できません。

- この機能は、通常の VPN クライアント（eBGP VPN クライアント）上では動作しません。

- ATTR_SET 内にパックされた属性は、iBGP CE 上の inbound route-policy で加えられた変更を反映し、指定した VRF の export route-policy で加えられた変更は反映しません。

- iBGP PE-CE ピアリング セッションで設定された同じ VPN の異なる VRF（つまり、異なる PE ルータ内）は、それぞれの VRF で異なるルート識別子（RD）を使用する必要があります。iBGP PE CE 機能は、RD 値が入力 VRF と出力 VRFで同じである場合は機能しません。

L3VPN iBGP PE-CE の設定

L3VPN iBGP PE-CE は、ネイバー、ネイバー グループ、またはセッション グループで有効にすることができます。L3VPN iBGP PE-CE を設定するには、次のステップを実行します。

始める前に

CE は、内部 BGP ピアである必要があります。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **vrf** *vrf-name*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# vrf blue
```

VRF インスタンスを設定します。

ステップ 4 **neighbor** *ip-address* **internal-vpn-client**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-vrf)# neighbor 10.0.0.0  internal-vpn-client
```

ルーティング情報を交換する相手の CE ネイバー デバイスを設定します。**neighbor internal-vpn-client** コマンドはVPN 属性セット内の iBGP-CE ネイバー パスをスタックします。

ステップ 5 **commit**

ステップ 6 **show bgp vrf** *vrf-name* **neighbors** *ip-address*

VRF CE ピアの iBGP PE-CE 機能が有効かどうかが表示されます。

ステップ 7 **show bgp** {**vpnv4** | **vpnv6** } **unicast rd**

L3VPN iBGP PE-CE が CE 上で有効になっている場合は、コマンドの出力に ATTR_SET 属性が表示されます。

例

例：L3VPN iBGP PE-CE の設定

次の例は、L3VPN iBGP PE-CE の設定方法を示しています。

```
R1(config-bgp-vrf-nbr)#neighbor 10.10.10.1 ?
. . .
 internal-vpn-client     Preserve iBGP CE neighbor path in ATTR_SET across VPN core
. . .
R1(config-bgp-vrf-nbr)#neighbor 10.10.10.1 internal-vpn-client
router bgp 65001
```

```
 bgp router-id 100.100.100.2
 address-family ipv4 unicast
address-family vpnv4 unicast
 !
 vrf ce-ibgp
  rd 65001:100
  address-family ipv4 unicast
  !
  neighbor 10.10.10.1
   remote-as 65001
   internal-vpn-client
```

次に、L3VPN iBGP PE-CE が CE ピアで有効になっている場合の **show bgp vrf** *vrf-name* **neighbors** *ip-address* コマンドの出力例を示します。

```
R1#show bgp vrf ce-ibgp neighbors 10.10.10.1
BGP neighbor is 10.10.10.1, vrf ce-ibgp
 Remote AS 65001, local AS 65001, internal link
 Remote router ID 100.100.100.1
  BGP state = Established, up for 00:00:19
  . . .
 Multi-protocol capability received
  Neighbor capabilities:
    Route refresh: advertised (old + new) and received (old + new)
    4-byte AS: advertised and received
    Address family IPv4 Unicast: advertised and received
CE attributes will be preserved across the core
  Received 2 messages, 0 notifications, 0 in queue
  Sent 2 messages, 0 notifications, 0 in queue
  . . .
```

次に、L3VPN iBGP PE-CE が CE ピアで有効になっている場合の **show bgp vpn4/vpn6 unicast rd** コマンドの出力例を示します。

```
BGP routing table entry for 1.1.1.0/24, Route Distinguisher: 200:300
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 10          10
Last Modified: Aug 28 13:11:17.000 for 00:01:00
Paths: (1 available, best #1)
  Advertised to update-groups (with more than one peer):
    0.2
Path #1: Received by speaker 0
  Advertised to update-groups (with more than one peer):
    0.2
  Local, (Received from a RR-client)
    20.20.20.2 from 20.20.20.2 (100.100.100.2)
      Received Label 24000
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
      not-in-vrf Received Path ID 0, Local Path ID 1, version 10
      Extended community: RT:228:237
     ATTR-SET [
         Origin-AS: 200
         AS-Path: 51320 52325 59744 12947 21969 50346 18204 36304 41213
 23906 33646
         Origin: incomplete
         Metric: 204
         Local-Pref: 234
         Aggregator: 304 34.3.3.3
         Atomic Aggregator
        Community: 1:60042 2:41661 3:47008 4:9280 5:39778 6:1069 7:15918
```

```
 8:8994 9:52701
10:10268 11:26276 12:8506 13:7131 14:65464 15:14304 16:33615 17:54991
18:40149 19:19401
        Extended community: RT:100:1 RT:1.1.1.1:1]
```

フロー タグの伝達

フロー タグ伝達機能では、ルート ポリシーとユーザ ポリシー間に相関関係を構築できます。BGP を使用したフロー タグ伝達では、AS 番号、プレフィックス リスト、コミュニティ文字列、および拡張コミュニティなどのルーティング属性に基づいてユーザ側でトラフィックをステアリングできます。フロー タグは論理数値識別子で、FIB ルックアップ テーブル内の FIB エントリのルーティング属性の 1 つとして RIB を通じて配布されます。フロー タグは、RPL からの「set」操作を使用してインスタンス化され、フロー タグ値に対してアクション（ポリシールール）が関連付けられている C3PL PBR ポリシーで参照されます。

フロー タグの伝達は次の場合に使用できます。

- 宛先 IP アドレス（コミュニティ番号を使用）またはプレフィックス（コミュニティ番号または AS 番号を使用）に基づいてトラフィックを分類する。
- カスタマー サイトのサービス レベル契約（SLA）に基づくサービス エッジに到達するパスのコストに合致する TE グループを選択する。
- SLA とそのクライアントに基づいて、特定のカスタマーにトラフィック ポリシー（TE グループの選択）を適用する。
- アプリケーション サーバまたはキャッシュ サーバにトラフィックを迂回させる。

フロー タグ伝達の制限

Border Gateway Protocol を使用した QoS ポリシー伝達（QPPB）とフロー タグ機能の併用については、いくつかの制限があります。次の作業を行います。

- ルート ポリシーには、「set qos-group」または「set flow-tag」のいずれかを使用できますが、prefix-set に両方は使用できません。
- qos-group と route policy flow-tag のルート ポリシーに重複するルートは使用できません。QPPB とフロー タグの機能は、それらが使用するルート ポリシーに重複するルートがない場合に限り、（同じインターフェイス上でも、異なるインターフェイス上でも）共存できます。
- ルート ポリシーとポリシー マップに qos-group と flow-tag を混在させて使用することはお勧めしません。

ソース ベースと宛先ベースのフロー タグ

ソースベースのフローのタグ機能では、着信パケットの発信元アドレスに割り当てられているフロータグに基づいてパケットを照合できます。一致した場合は、このポリシーでサポートされている PBR アクションを適用できます。

送信元と送信先ベースのフロー タグの設定

指定したインターフェイスにフロータグを適用するには、このタスクを実行します。パケットは、着信パケットの発信元アドレスに割り当てられているフロー タグに基づいて照合されます。

（注） インターフェイスでQPPBとフロータグ機能の両方を同時にイネーブルにすることはできません。

手順

ステップ 1 **configure**

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

例：

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

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

ステップ 3 **ipv4 | ipv6 bgp policy propagation input flow-tag{destination | source}**

例：

```
RP/0/RP0/cpu 0: router(config-if)# ipv4 bgp policy propagation input flow-tag source
```

送信元または送信先の IP アドレスのフロー タグ ポリシーの伝達をインターフェイスで有効にします。

ステップ 4 **commit**

例

次の show コマンドは、ルータに適用された RBP ポリシーを使用して出力を表示します。

```
show running-config interface gigabitEthernet 0/0/0/12
Thu Feb 12 01:51:37.820 UTC
interface GigabitEthernet0/0/0/12
 service-policy type pbr input flowMatchPolicy
 ipv4 bgp policy propagation input flow-tag source
 ipv4 address 192.5.1.2 255.255.255.0
!

RP/0/RSP0/CPU0:ASR9K-0#show running-config policy-map type pbr flowMatchPolicy
Thu Feb 12 01:51:45.776 UTC
policy-map type pbr flowMatchPolicy
 class type traffic flowMatch36
```

```
  transmit
 !
 class type traffic flowMatch38
  transmit
 !
 class type traffic class-default
 !
 end-policy-map
!

RP/0/RSP0/CPU0:ASR9K-0#show running-config class-map type traffic flowMatch36
Thu Feb 12 01:52:04.838 UTC
class-map type traffic match-any flowMatch36
 match flow-tag 36
 end-class-map
!
```

BGP キーチェーン

BGP キーチェーンを使用すると、2 つの BGP ピア間のキーチェーン認証がイネーブルになります。BGP のエンドポイントは、どちらも draft-bonica-tcp-auth-05.txt を順守する必要があり、一方のエンドポイントのキーチェーンと、もう一方のエンドポイントのパスワードは機能しません。

BGP では、認証にこのキーチェーンを使用して、ヒットレス キー ロールオーバーを実装できます。キー ロールオーバーの仕様は時間に基づいているため、ピア間で時計のずれがあるとロールオーバーのプロセスに影響します。許容値の指定を設定できるため、承認時間枠をその分だけ（前後に）拡張できます。この承認時間枠により、アプリケーション（ルーティングプロトコルおよび管理プロトコルなど）のヒットレス キー ロールオーバーが容易になります。

キーのロール オーバーは、エンドポイントでのキーチェーン設定の不一致が原因でセッショントラフィック（送信または受信）で使用する共通のキーがない場合を除き、BGPセッションには影響しません。

BGP のキーチェーンの設定

キーチェーンは、さまざまな MAC 認証アルゴリズムをサポートして安全な認証を実現し、円滑なキー ロールオーバーを実装します。BGP のキーチェーンを設定するには、次の作業を実行します。このタスクはオプションです。

（注） ネイバー グループまたはセッション グループのキーチェーンが設定されている場合、そのグループを使用するネイバーはキーチェーンを継承します。あるネイバーのために特別に設定されたコマンドの値は、継承された値を上書きします。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.168.40.24
```

BGPルーティングのためにルータをネイバーコンフィギュレーションモードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

ステップ 4 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 2002
```

ネイバーを作成し、リモート自律システム番号を割り当てます。

ステップ 5 **keychain** *name*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# keychain kych_a
```

キーチェーンに基づく認証を設定します。

ステップ 6 **commit**

マスター キー タプル設定

この機能は、TCP MD5 オプションを置き換える TCP 認証オプション（TCP-AO）を指定します。TCP-AO は、以下を提供するメッセージ認証コード（MAC）を使用します。

- 長時間の TCP 接続のリプレイに対する保護
- TCP MD5 以外の TCP 接続でのセキュリティ アソシエーションの詳細
- 他のシステムや操作の変更を最小限に抑えた多数の MAC

TCP-AO は、マスター キー タプル（MKT）設定と互換性があります。TCP-AO は、接続の繰り返しインスタンスで同じ MKT を使用する場合も接続を保護します。TCP-AO は、MKT から導出されたトラフィック キーを使用して接続を保護し、エンドポイント間の変更を調整します。

（注） TCPAO と TCP MD5 を同時に使用することはできません。TCP-AO は IPv6 をサポートしており、TCP MD5 の交換に関して提案される要件と完全に互換性があります。

シスコでは、次の設定を介して MKT 設定を提供しています。

- キーチェーン設定
- tcp ao キーチェーン設定

システムは、キーチェーンの下にある 「key_id」 などの各キーを MKT として変換します。キーチェーン設定には、秘密、ライフタイム、アルゴリズムなどの設定の一部が含まれます。「tcp ao キーチェーン」 モードには、MKT 用の TCP AO 固有の設定（send_id および receive_id）が含まれています。

キーチェーン設定

設定時の注意事項

設定を正常に実行するには、設定に関する注意事項に従ってください。

- Send_ID と Receive_ID の両方で許可されている値の範囲は 0 ～ 255 です。
- アプリケーション ネイバーには、1 つのキーチェーンのみをリンクできます。
- 同一のキーチェーンで、ライフタイムが重複しているキーの下に同じ send_id キーを再度設定すると、設定を修正するまで古いキーは使用できなくなります。
- 次のシナリオでは、システムから警告メッセージが送信されます。
  - Send_ID または Receive_ID が変更された場合。
  - 対応するキーが現在アクティブで、一部の接続で使用されている場合。
- BGP ネイバーは、次のいずれかの認証オプションのみを使用できます。
  - MD5
  - EA
  - AO

（注） これらのオプションのいずれかを設定すると、設定時にシステムによって他の認証オプションが拒否されます。

TCP AO BGP ネイバーの設定時の注意事項

設定時の注意事項は次のとおりです。

- key_id を使用する必要があるライフタイムを指定して、key_id ですべての必要な設定（key_string、MAC_algorithm、send_lifetime、accept_lifetime、send_id、receive_id）を行います。
- ピア側で、一致する MKT をまったく同じライフタイムで設定します。
- キーチェーンキーが tcp-ao にリンクされた後は、キーのコンポーネントを変更しないでください。TCP に別のキーの使用を検討させる場合は、そのキーを動的に設定できます。送信ライフタイムの「start-time」に基づいて、TCP AO はキーを使用します。
- （キーチェーンの下にある）key_id の Send_ID と Receive_ID は、ライフタイム範囲が同じである必要があります（たとえば、send-lifetime==accept-lifetime）。

  TCP は send-lifetime の期限切れのみを考慮して次のアクティブ キーに移行します。accept-lifetime はまったく考慮されません。
- 特定のキーの send-lifetime でカバーされる send-lifetime を別のキーに設定しないでください。

  たとえば、既存のキーの send-lifetime が「04:00:00 November 01, 2017 07:00:00 November 01, 2017」に設定されている場合、ユーザが別のキーの send-lifetime を「05:00:00 November 01, 2017 06:00:00 November 01, 2017」に設定すると、接続フラップが発生する可能性があります。

  新しいキーが期限切れになると、TCP AO は古いキーに戻そうとします。ただし、新しいキーがすでに期限切れになっている場合、TCP AO はこのキーを使用できないため、セグメント損失や接続フラップが発生する可能性があります。
- 重複する 2 つのキー間の重複時間は 15 分以上に設定します。TCP は期限が切れたキーを使用しないため、そのキーを使用した不適切なセグメントはドロップされます。
- 簡素化のために、key_id に設定する send_id と receive_id を同一にすることを推奨します。
- TCP には、キーチェーンに含まれるキーチェーンおよびキーの数に関する制限はありません。システムでは 4000 を超えるキーチェーンはサポートされません。4000 を超えると、予期しない動作が発生する可能性があります。

キーチェーン設定

```
key chain <keychain_name>
   key <key_id>
      accept-lifetime <start-time> <end-time>
      key-string <master-key>
      send-lifetime <start-time> <end-time>
      cryptographic-algorithm <algorithm>
   !
!
```

TCP 設定

TCP は、各キーチェーンの key_id ごとに SendID および ReceiveID を指定する新しい tcp ao サブモードを提供します。

```
tcp ao
    keychain <keychain_name1>
        key-id <key_id> send_id <0-255> receive_id <0-255>
        !
```

例：

```
tcp ao
 keychain bgp_ao
  key 0 SendID 0 ReceiveID 0
  key 1 SendID 1 ReceiveID 1
  key 2 SendID 3 ReceiveID 4
 !
 keychain ldp_ao
  key 1 SendID 100 ReceiveID 200
  key 120 SendID 1 ReceiveID 1
 !
```

BGP 設定

BGP などのアプリケーションは、tcp-ao キーチェーンと、ネイバーごとに使用する関連情報を提供します。次に、tcp-ao キーチェーンごとのオプション設定を示します。

- include-tcp-options
- accept-non-ao-connections

```
router bgp <AS-number>
neighbor <neighbor-ip>
  remote-as <remote-as-number>
  ao <keychain-name> include-tcp-options enable/disable  <accept-ao-mismatch-connections>
!
```

XML 設定

BGP XML

TCP-AO XML

```
<?xml version="1.0" encoding="UTF-8"?>
<Request>
 <Set>
  <Configuration>
   <IP_TCP>
    <AO>
      <Enable>
       true
      </Enable>
      <KeychainTable>
        <Keychain>
         <Naming>
          <Name> bgp_ao_xml </Name>
         </Naming>
         <Enable>
          true
         </Enable>
          <KeyTable>
           <Key>
            <Naming>
             <KeyID> 0 </KeyID>
            </Naming>
             <SendID> 0 </SendID>
             <ReceiveID> 0 </ReceiveID>
           </Key>
          </KeyTable>
        </Keychain>
      </KeychainTable>
    </AO>
   </IP_TCP>
  </Configuration>
 </Set>
 <Commit/>
</Request>
```

BGP ノンストップ ルーティング

ボーダー ゲートウェイ プロトコル（BGP）のノンストップ ルーティング（NSR）とステートフル スイッチオーバー（SSO）機能を使用すると、すべての bgp ピアリングで BGP 状態を維持し、サービスを中断させるおそれのあるイベントの実行中にも連続的なパケット転送を行えるようになります。NSR の下では、サービスを中断するおそれのあるイベントは、ピア ルータに表示されません。プロトコル セッションは中断されず、ルーティング ステートはプロセスの再起動とスイッチオーバーをまたがって維持されます。

BGP ノンストップ ルーティング リファレンス（145 ページ） で詳細情報を参照してください。

BGP ノンストップ ルーティングの設定

BGP ノンストップ ルーティング（BGP NSR）はデフォルトで有効になっています。BGP NSR が無効になっている場合は、**no nsr disable** コマンドを使用して BGP NSR を有効に戻します。

（注） 場合によっては、一部またはすべての bgp セッションが NSR 対応ではない可能性があります。この場合も、`show redundancy` コマンドで bgp セッションが NSR 対応であると示されることがあります。そのため、`show bgp sessions` コマンドを使用して、bgp nsr の状態を確認することを推奨します。

BGP ノンストップ ルーティングの無効化

BGP ノンストップ ルーティング（NSR）を無効にするには、次のタスクを実行します。

手順

ステップ 1 **configure**

ステップ 2 **router bgp** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config)# router bgp 120
```

BGP ルーティング プロセスを設定するため、BGP AS 番号を指定して BGP コンフィギュレーション モードを開始します。

ステップ 3 **nsr disable**

例：

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

BGP ノンストップ ルーティングを無効にします。

ステップ 4 **commit**

BGP ノンストップ ルーティングの無効化：例

次に、BGP NSR をディセーブルにする例を示します。

```
configure
router bgp 120
no nsr
end
```

BGP ノンストップ ルーティングの再有効化

BGP ノンストップ ルーティング（NSR）が無効になっている場合、次のステップを使用して BGP NSR を有効にします。

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 120
```

BGP ルーティング プロセスを設定するため、BGP AS 番号を指定して BGP コンフィギュレーション モードを開始します。

ステップ 3 **no nsr disable**

例：

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

BGP ノンストップ ルーティングを有効にします。

ステップ 4 **commit**

BGP ノンストップ ルーティングの再有効化：例

次に、BGP NSR をイネーブルにする例を示します。

```
configure
router bgp 120
nsr
end
```

累積内部ゲートウェイ プロトコル属性

累積内部ゲートウェイ プロトコル（AiGP）属性は、オプションで非推移的な BGP パス属性です。AiGP 属性の属性タイプ コードは、IANA によって割り当てられます。AiGP 属性の値フィールドは、タイプ、長さ、値（TLV）の要素として定義されます。AiGP TLV には、累積 IGP メトリックが含まれます。

AiGP 機能は 3107 ネットワークに必要であり、パスに関連付けられた距離を計算する現在の OSPF の動作をシミュレートします。OSPF/LDP では、プレフィックスおよびラベル情報をロー

カル領域だけに入れて伝送します。次に、BGP では、エリア境界にある BGP にルートを再配布することにより、すべてのリモートエリアにプレフィックスおよびラベルを伝送します。次に、ルートおよびラベルが、LSP を使用してアドバタイズされます。ルートのネクストホップはローカル ルータに対する各 ABR で変更されます。これによって、エリア境界を越えて OSPF ルートをリークする必要がなくなります。各コア リンクで使用可能な帯域幅が OSPF コストにマップされます。したがって、BGP では、各 PE 間でこのコストを正しく伝送する必要があります。この機能は、AiGP を使用して実現されています。

AiGP によるプレフィックスの生成

AiGP メトリックを使用したルートの生成を設定するには、次の作業を実行します。

始める前に

Accumulated Interior Gateway Protocol（AiGP）メトリックを使用したルートの生成は設定により制御されます。次の条件を満たす再配布ルートに AiGP 属性が付加されます。

- AiGP でルートを再配布するプロトコルがイネーブルに設定されている。
- このルートは、ボーダー ゲートウェイ プロトコル（BGP）に再配布された Interior Gateway Protocol（iGP）ルートです。AiGP 属性に割り当てられた値はルートの iGP ネクスト ホップの値か、または route-policy によって設定された値です。
- このルートは BGP に再配布されたスタティック ルートです。割り当てられた値はルートのネクスト ホップの値か、route-policy によって設定された値です。
- このルートはネットワーク ステートメントによって BGP にインポートされます。割り当てられた値はルートのネクスト ホップの値か、route-policy によって設定された値です。

手順

ステップ 1 **configure**

ステップ 2 **route-policy** *aigp_policy*

例：

```
RP/0/RP0/cpu 0: router(config)# route-policy aip_policy
```

ルート ポリシー コンフィギュレーション モードを開始してルート ポリシーを設定します。

ステップ 3 **set aigp-metricigp-cost**

例：

```
RP/0/RP0/cpu 0: router(config-rpl)# set aigp-metric igp-cost
```

内部ルーティング プロトコル コストを aigp メトリックとして設定します。

ステップ 4 **exit**

例：

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

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 5 **router bgp** *as-number*

例：

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

BGP AS 番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 6 **address-family** {**ipv4** | **ipv6**} **unicast**

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

ステップ 7 **redistribute ospf** *osp* **route-policy** *plcy_name***metric** *value*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)#redistribute ospf osp route-policy aigp_policy
metric 1
```

OSPF への AiBGP メトリックの再配布を許可します。

ステップ 8 **commit**

AiGP によるプレフィックスの生成：例

次に、AiGP メトリック属性を使用してプレフィックスを生成するための設定例を示します。

```
route-policy aigp-policy
  set aigp-metric 4
  set aigp-metric igp-cost
end-policy
!
router bgp 100
 address-family ipv4 unicast
  network 10.2.3.4/24 route-policy aigp-policy
  redistribute ospf osp1 metric 4 route-policy aigp-policy
 !
!
end
```

BGP Accept Own の設定

BGP Accept Own 機能を使用すると、自動送信 VPN ルート（BGP スピーカーがルート リフレクタ（RR）から受信するルート）を処理できるようになります。「自動送信」ルートは、ス

ピーカー自体によって最初にアドバタイズされたルートです。BGPプロトコル（RFC4271）に従って、BGPスピーカーは、スピーカー自体によって送信されたアドバタイズメントを拒否します。ただし、BGP Accept Own メカニズムを使用すると、プレフィックスの特定の属性を変更するルートリフレクタから反映された場合に、ルータは自身がアドバタイズしたプレフィックスを受け入れることが可能になります。ACCEPT-OWN と呼ばれる特別なコミュニティがルート リフレクタによってプレフィックスに付加されます。これは ORIGINATOR_ID および NEXTHOP/MP_REACH_NLRI チェックをバイパスするための受信側ルータに対する信号です。通常、BGP スピーカーは自動送信されたプレフィックスを自動送信チェック（ORIGINATOR_ID、NEXTHOP/MP_REACH_NLRI）によって検出し、受信した更新をドロップします。ただし、更新に Accept Own コミュニティがあれば、BGP スピーカーはそのルートを処理します。

BGP Accept Own の応用例の 1 つは、MPLS VPN ネットワーク内のエクストラネットの自動設定です。エクストラネットの設定では、ある VRF にあるルートは同じ PE の別の VRF にインポートされます。通常、エクストラネットのメカニズムでは、別の VRF からのプレフィックスのインポートを制御するために、エクストラネット VRF のインポート RT またはインポートポリシーを編集する必要があります。ただし、Accept Own 機能を使用すると、ルート リフレクタは、PE で設定変更することなく、その制御をアサートできます。このように Accept Own 機能によって、異なる VRF 間でのルートのインポートの制御を集中管理できます。

（注） BGP Accept Own 機能は、ネイバー コンフィギュレーション モードの VPNv4 および VPNv6 アドレス ファミリ向けにのみサポートされています。

BGP Accept Own を設定するには、次の作業を実行します。

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 100
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)#neighbor 10.1.2.3
```

BGPルーティングのためにルータをネイバーコンフィギュレーションモードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

ステップ 4 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)#remote-as 100
```

ネイバーにリモート自律システム番号を割り当てます。

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

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)#update-source Loopback0
```

ネイバーでセッションを形成するとき、特定のインターフェイスからのプライマリ IP アドレスをローカル アドレスとしてセッションで使用できます。

ステップ 6 **address-family** {*vpnv4 unicast* | *vpnv6 unicast*}

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)#address-family vpnv6 unicast
```

アドレス ファミリを VPNv4 または IPv6 として指定し、ネイバー アドレス ファミリのコンフィギュレーション モードを開始します。

ステップ 7 **accept-own** [**inheritance-disable**]

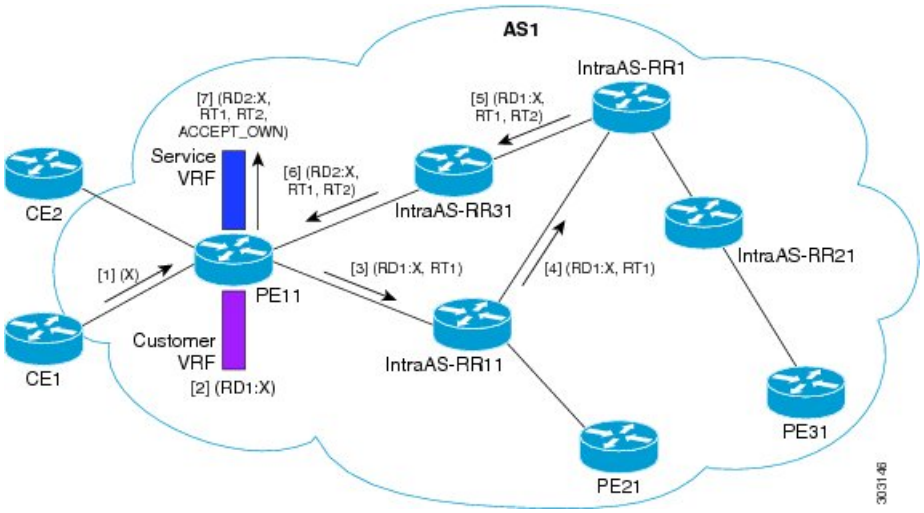
例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)#accept-own
```

Accept_Own コミュニティが含まれる自動送信 VPN ルートの処理をイネーブルにします。

「Accept Own」設定をディセーブルにし、親コンフィギュレーションから「Accept Own」が継承されないようにするには、 **inheritance-disable** キーワードを使用します。

BGP Accept Own の設定：例



この設定例の内容は次のとおりです。

- PE11 にカスタマー VRF とサービス VRF が設定されています。

- OSPF は IGP として使用されます。
- VPNv4 ユニキャストおよび VPNv6 ユニキャストのアドレス ファミリが PE ネイバーと RR ネイバーとの間でイネーブルになっており、IPv4 および IPv6 が PE ネイバーと CE ネイバーとの間でイネーブルになっています。

Accept Own の設定は次のように動作します。

1. CE1 がプレフィックス X を発信します。
2. プレフィックス X は、カスタマー VRF に (RD1:X) として設定されています。
3. プレフィックス X は IntraAS-RR11 に (RD1:X, RT1) としてアドバタイズされます。
4. IntraAS-RR11 が InterAS-RR1 に X を (RD1:X, RT1) としてアドバタイズします。
5. InterAS-RR1 はインバウンドのプレフィックス X とアウトバウンドの ACCEPT_OWN コミュニティに RT2 を付加し、IntraAS-RR31 にプレフィックス X をアドバタイズします。
6. IntraAS-RR31 が PE11 に X をアドバタイズします。
7. PE11 は X をサービス VRF に (RD2:X,RT1, RT2, ACCEPT_OWN) としてインストールします。

次に、BGP Accept Own を PE ルータに設定する例を示します。

```
router bgp 100
 neighbor 45.1.1.1
   remote-as 100
   update-source Loopback0
   address-family vpnv4 unicast
    route-policy pass-all in
    accept-own
    route-policy drop_111.x.x.x out
   !
   address-family vpnv6 unicast
    route-policy pass-all in
    accept-own
    route-policy drop_111.x.x.x out
   !
  !
```

次の例は、BGP Accept Own のための InterAS-RR の設定を示しています。

```
router bgp 100
 neighbor 45.1.1.1
  remote-as 100
  update-source Loopback0
  address-family vpnv4 unicast
   route-policy rt_stitch1 in
   route-reflector-client
   route-policy add_bgp_ao out
  !
  address-family vpnv6 unicast
   route-policy rt_stitch1 in
   route-reflector-client
   route-policy add_bgp_ao out
  !
 !
```

```
extcommunity-set rt cs_100:1
  100:1
end-set
!
extcommunity-set rt cs_1001:1
  1001:1
end-set
!
route-policy rt_stitch1
  if extcommunity rt matches-any cs_100:1 then
    set extcommunity rt cs_1000:1 additive
 endif
end-policy
!
route-policy add_bgp_ao
  set community (accept-own) additive
end-policy
!
```

BGP リンクステート

BGP リンクステート（LS）は、BGP を介して内部ゲートウェイ プロトコル（IGP）リンクステートデータベースを伝えるために定義されたアドレス ファミリ識別子（AFI）およびサブアドレス ファミリ識別子（SAFI）です。BGP LS は、ネットワーク トポロジ情報をトポロジ サーバおよびアプリケーション層トラフィック最適化（ALTO）サーバに提供します。BGP LS では、集約、情報の非表示、および抽象化に対するポリシー ベースの制御が可能です。BGP LS は、IS-IS および OSPFv2 をサポートしています。

（注） IGP は、リモート ピアからの BGP LS データを使用しません。BGP は、ルータの他のコンポーネントに受信した BGP LS データをダウンロードしません。

BGP リンク ステートの設定

BGP リンクステート（LS）情報を BGP ネイバーと交換するには、次のステップを実行します。

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 100
```

BGP AS 番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 10.0.0.2
```

CE ネイバーを設定します。ip-address 引数は、プライベート アドレスである必要があります。

ステップ 4 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 1
```

CE ネイバーのリモート AS を設定します。

ステップ 5 **address-family link-state link-state**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# address-family link-state link-state
```

BGP リンクステート情報を指定されたネイバーに配布します。

ステップ 6 **commit**

ドメイン識別子の設定

固有識別子 4 オクテット ASN を設定するには、次のステップを実行します。

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 100
```

BGP AS 番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **address-family link-state link-state**

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# address-family link-state link-state
```

アドレスファミリ リンクステート コンフィギュレーション モードを開始します。

ステップ 4 **domain-distinguisher** *unique-id*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)# domain-distinguisher 1234
```

固有識別子 4 オクテット ASN を設定します。範囲は 1 ～ 4294967295 です。

ステップ 5 **commit**

BGP パーマネント ネットワーク

BGP パーマネント ネットワーク機能は、BGP 経由のスタティック ルーティングをサポートしています。（ルート ポリシーで識別された）IPv4 または IPv6 宛先への BGP ルートは、管理用に作成して、BGP ピアに選択的にアドバタイズできます。これらのルートは、管理上削除されるまでルーティング テーブルに残ります。パーマネント ネットワークは、プレフィックスのセットを永続的なものとして定義するために使用されます。つまり、プレフィックスのセットのアップストリームにおいて BGP のアドバタイズメントまたは取り消しは 1 回しかありません。プレフィックス セットの各ネットワークに対し、BGP 固定パスが作成され、優先度はそのピアから受信される他の BGP パスよりも低く扱われます。BGP 固定パスが最適パスである場合は RIB にダウンロードされます。

グローバル アドレス ファミリ コンフィギュレーション モードの **permanent-network** コマンドは、ルートポリシーを使用して固定パスが設定されるプレフィックス（ネットワーク）のセットを識別します。ネイバー アドレスファミリ コンフィギュレーション モードの **advertise permanent-network** コマンドは、固定パスをアドバタイズする必要があるピアの識別に使用されます。別の最適パスが使用可能であっても、固定パスは常にアドバタイズ パーマネント ネットワーク設定を持つピアにアドバタイズされます。固定パスは、固定パスを受信するように設定されていないピアにはアドバタイズされません。

パーマネント ネットワーク機能は、デフォルトの仮想ルーティングおよび転送（VRF）下の IPv4 ユニキャストおよび IPv6 ユニキャスト アドレス ファミリ内のプレフィックスのみをサポートします。

制約事項

次の制限は、パーマネント ネットワークの設定時に適用されます。

- パーマネント ネットワーク プレフィックスは、グローバル アドレス ファミリでルート ポリシーによって指定する必要があります。

- グローバル アドレス ファミリ コンフィギュレーション モードでルート ポリシーを使用してパーマネント ネットワークを構成し、それをネイバー アドレス ファミリ コンフィギュレーション モードで設定する必要があります。
- パーマネント ネットワーク設定を削除する場合は、ネイバー アドレス ファミリ コンフィギュレーション モードの設定を削除してから、グローバル アドレス ファミリ コンフィギュレーション モードから削除します。

BGP パーマネント ネットワークの設定

BGP パーマネント ネットワークを設定するには、次のタスクを実行します。パーマネント ネットワーク（パス）が設定されるプレフィックス（ネットワーク）のセットを識別するには、少なくとも 1 つのルート ポリシーを設定する必要があります。

手順

ステップ 1 **configure**

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

例：

```
RP/0/RP0/cpu 0: router(config)# prefix-set PERMANENT-NETWORK-IPv4
RP/0/RP0/cpu 0: router(config-pfx)# 1.1.1.1/32,
RP/0/RP0/cpu 0: router(config-pfx)# 2.2.2.2/32,
RP/0/RP0/cpu 0: router(config-pfx)# 3.3.3.3/32
RP/0/RP0/cpu 0: router(config-pfx)# end-set
```

プレフィックス セット コンフィギュレーション モードを開始し、連続したビット セットと非連続のビット セットに対しプレフィックス セットを定義します。

ステップ 3 **exit**

例：

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

プレフィックス セット コンフィギュレーション モードを終了し、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 4 **route-policy** *route-policy-name*

例：

```
RP/0/RP0/cpu 0: router(config)# route-policy POLICY-PERMANENT-NETWORK-IPv4
RP/0/RP0/cpu 0: router(config-rpl)# if destination in PERMANENT-NETWORK-IPv4 then
RP/0/RP0/cpu 0: router(config-rpl)# pass
RP/0/RP0/cpu 0: router(config-rpl)# endif
```

ルート ポリシーを作成し、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。このモードではルート ポリシーを定義できます。

ステップ 5 **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 6 **router bgp** *as-number*

例：

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

自律システム番号を指定して、BGP コンフィギュレーション モードを開始します。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

ステップ 8 **permanent-network route-policy** *route-policy-name*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-af)# permanent-network route-policy
POLICY-PERMANENT-NETWORK-IPv4
```

ルート ポリシーで定義されているプレフィックスのセットに対しパーマネント ネットワーク（パス）を設定します。

ステップ 9 **commit**

ステップ 10 **show bgp {ipv4 | ipv6} unicast** *prefix-set*

例：

```
RP/0/RP0/cpu 0: routershow bgp ipv4 unicast
```

（オプション）プレフィックス セットが BGP でパーマネント ネットワークであるかどうかを表示します。

パーマネント ネットワークのアドバタイズ

固定パスがアドバタイズされる必要があるピアを識別するには、このタスクを実行します。

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 100
```

自律システム番号を指定して、BGP コンフィギュレーション モードを開始します。

ステップ 3 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 10.255.255.254
```

BGPルーティングのためにルータをネイバーコンフィギュレーションモードにして、ネイバーの IP アドレスを BGP ピアとして設定します。

ステップ 4 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 4713
```

ネイバーをリモート自律システム番号に割り当てます。

ステップ 5 **address-family { ipv4 | ipv6 } unicast**

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

ステップ 6 **advertise permanent-network**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# advertise permanent-network
```

パーマネント ネットワーク（パス）がアドバタイズされるピアを指定します。

ステップ 7 **commit**

ステップ 8 **show bgp {ipv4 | ipv6} unicast neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: routershow bgp ipv4 unicast neighbor 10.255.255.254
```

（オプション）ネイバーが BGP パーマネント ネットワークを受信できるかどうかを表示します。

BGP 不等コストの連続ロード バランシングの有効化

手順

	コマンドまたはアクション	目的
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 120`	自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。
ステップ 3	**address-family** { **ipv4** | **ipv6** } **unicast** 例： `RP/0/RP0/cpu 0: router(config-bgp)# address-family ipv4 unicast`	IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。 このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。
ステップ 4	**maximum-paths** { **ebgp** | **ibgp** | **eibgp** } *maximum* [**unequal-cost**] 例： `RP/0/RP0/cpu 0: router(config-bgp-af)# maximum-paths ebgp 3`	BGP によりルーティング テーブルにインストールされるパラレル ルートの最大数を設定します。 • **ebgp** *maximum* ：マルチパスに eBGP パスのみを考慮します。 • **ibgp** *maximum* [**unequal-cost**]：iBGP 学習パス間でのロードバランシングを考慮します。 • **eibgp** *maximum* ：eBGP および iBGP 学習パスの両方のロード バランシ

	コマンドまたはアクション	目的
		ングを考慮します。eiBGP は常に不等コスト ロード バランシングを実行します。 eiBGP が適用されると eBGP ロードバランシングまたは iBGP ロードバランシングは設定できませんが、eBGP ロードバランシングと iBGP ロードバランシングは共存できます。
ステップ 5	**exit** 例： `RP/0/RP0/cpu 0: router(config-bgp-af)# exit`	現在のコンフィギュレーション モードを終了します。
ステップ 6	**neighbor** *ip-address* 例： `RP/0/RP0/cpu 0: router(config-bgp)# neighbor 10.0.0.0`	CE ネイバーを設定します。 *ip-address* 引数は、プライベート アドレスにする必要があります。
ステップ 7	**dmz-link-bandwidth** 例： `RP/0/RP0/cpu 0: router(config-bgp-nbr)# dmz-link-bandwidth`	eBGP および iBGP ネイバーへのリンクのために、非武装地帯（DMZ）リンク帯域幅拡張コミュニティを開始します。
ステップ 8	**commit**	

不等コストの連続ロード バランシングに対する DMZ リンク帯域幅

不等コストの連続ロード バランシングに対する非武装地帯（DMZ）リンク帯域幅機能では、DMZ リンク帯域幅を使用して、ローカル ノード上の連続プレフィックスに対する不等コスト ロード バランシングをサポートします。BGP ネイバー コンフィギュレーション モードで dmz-link-bandwidth コマンドを使用し、インターフェイス コンフィギュレーション モードで bandwidth コマンドを使用して、不等ロード バランシングを実行します。

マルチプロトコル内部 BGP（MP-iBGP）セッション（IPv4 または VPNv4）を介した、リモート PE への PE ルータのアップデートにリンク帯域幅拡張コミュニティが含まれている場合、**maximum-paths** コマンドが有効になっていれば、リモート PE が自動的にロード バランシングを実行します。

（注） 不等コストの連続ロード バランシングは、最大で 8 つのパスに対してのみ行われます。

BGP 不等コストの連続ロード バランシングの有効化

手順

	コマンドまたはアクション	目的
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 120`	自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。
ステップ 3	**address-family** { **ipv4** | **ipv6** } **unicast** 例： `RP/0/RP0/cpu 0: router(config-bgp)# address-family ipv4 unicast`	IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。 このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。
ステップ 4	**maximum-paths** { **ebgp** | **ibgp** | **eibgp** } *maximum* [**unequal-cost**] 例： `RP/0/RP0/cpu 0: router(config-bgp-af)# maximum-paths ebgp 3`	BGP によりルーティング テーブルにインストールされるパラレル ルートの最大数を設定します。 • **ebgp** *maximum* ：マルチパスに eBGP パスのみを考慮します。 • **ibgp** *maximum* [**unequal-cost**]：iBGP 学習パス間でのロード バランシングを考慮します。 • **eibgp** *maximum* ：eBGP および iBGP 学習パスの両方のロード バランシングを考慮します。eiBGP は常に不等コスト ロード バランシングを実行します。 eiBGP が適用されると eBGP ロードバランシングまたは iBGP ロードバランシングは設定できませんが、eBGP ロードバランシングと iBGP ロードバランシングは共存できます。
ステップ 5	**exit** 例：	現在のコンフィギュレーション モードを終了します。

	コマンドまたはアクション	目的
	`RP/0/RP0/cpu 0: router(config-bgp-af)# exit`	
ステップ 6	**neighbor** *ip-address* 例： `RP/0/RP0/cpu 0: router(config-bgp)# neighbor 10.0.0.0`	CE ネイバーを設定します。 *ip-address* 引数は、プライベート アドレスにする必要があります。
ステップ 7	**dmz-link-bandwidth** 例： `RP/0/RP0/cpu 0: router(config-bgp-nbr)# dmz-link-bandwidth`	eBGP および iBGP ネイバーへのリンクのために、非武装地帯（DMZ）リンク帯域幅拡張コミュニティを開始します。
ステップ 8	**commit**	

EBGP ピア上の DMZ リンク帯域幅

非武装ゾーン（DMZ）リンク帯域幅拡張コミュニティは、オプションの非遷移属性です。したがって、デフォルトでは eBGP ピアにはアドバタイズされず、iBGP ピアのみにアドバタイズされます。この拡張コミュニティは、マルチパスを介したロード バランシング用です。ただし、Cisco IOS-XR は、eBGP ピアへの DMZ リンク帯域幅のアドバタイズと、eBGP ピアによる DMZ リンク帯域幅の受信を可能にします。また、この機能は帯域幅をそのまま送信するか、またはすべての出力リンク上の累積帯域幅を取得して上流側の eBGP ピアにアドバタイズするオプションもユーザに提供します。

コミュニティを eBGP ピアに送信するには、**ebgp-send-community-dmz** コマンドを使用します。デフォルトでは、最適パスに関連付けられたリンク帯域幅拡張コミュニティ属性が送信されます。

cumulative キーワードを使用すると、リンク帯域幅拡張コミュニティの値が、すべての出力マルチパスのリンク帯域幅値の合計に設定されます。一部のパスにその属性がないなど、マルチパスの DMZ リンク帯域幅の値がわからない場合は、そのノードでは不等コスト ロードバランシングは実行されません。ただし、既知の DMZ リンク帯域幅値の合計を計算して eBGP ピアに送信します。

eBGP ピアからコミュニティを受信するには、**ebgp-recv-community-dmz** コマンドを使用します。

（注） **ebgp-send-community-dmz** コマンドと **ebgp-recv-community-dmz** コマンドは、ネイバー、ネイバーグループ、およびセッショングループ コンフィギュレーション モードで設定できます。

複数の自律システム間でマルチパスを処理するには、**bgp bestpath as-path multipath-relax** および **bgp bestpath as-path ignore** コマンドを使用します。

eBGP ピアを介した DMZ リンク帯域幅拡張コミュニティの送受信

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 100
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 10.1.1.1
```

BGP ルーティング セッションを設定するには、ネイバー コンフィギュレーション モードを開始します。

ステップ 4 **ebgp-send-extcommunity-dmz** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# ebgp-send-extcommunity-dmz
```

DMZ リンク帯域幅拡張コミュニティを eBGP ネイバーに送信します。

（注） リンク帯域幅拡張コミュニティの値をすべての出力マルチパスのリンク帯域幅値の合計に設定するには、このコマンドで **cumulative** キーワードを使用します。

ステップ 5 **exit** *ip-address*

例：

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

ネイバー コンフィギュレーション モードを終了し、BGP コンフィギュレーション モードを開始します。

ステップ 6 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.16.0.1
```

BGP ルーティング セッションを設定するには、ネイバー コンフィギュレーション モードを開始します。

ステップ 7 **ebgp-recv-extcommunity-dmz**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# ebgp-recv-extcommunity-dmz
```

eBGP ネイバーへの DMZ リンク帯域幅拡張コミュニティを受け取ります。

ステップ 8 **exit** *ip-address*

例：

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

ネイバー コンフィギュレーション モードを終了し、BGP コンフィギュレーション モードを開始します。

DMZ リンク帯域幅：例

次に、ルータ R1 が DMZ リンク帯域幅拡張コミュニティを eBGP ピア接続を介してルータ R2 に送信する例を示します。

```
R1: sending router
------------------
neighbour 10.3.3.3
  remote-as 2
  ebgp-send-extcommunity-dmz
  address-family ipv4 unicast
   route-policy pass in
   route-policy pass out
  !

R2: Receiving router
--------------------
neighbor 192.0.2.1
  remote-as 3
  ebgp-recv-extcommunity-dmz
  address-family ipv4 unicast
   route-policy pass in
  !
route-policy pass out
!
```

次に、送信側（R1）ルータの DMZ リンク帯域幅設定を表示する設定の例を示します。

```
RP/0/RP0/CPU0:router)# show bgp ipv4 unicast 10.1.1.1/32 detail

Path #1: Received by speaker 0
  Flags: 0x4000000001040003, import: 0x20
  Advertised to update-groups (with more than one peer):
    0.4
  Advertised to peers (in unique update groups):
    20.0.0.1
  3
    11.1.0.2 from 11.1.0.2 (11.1.0.2)
      Origin incomplete, metric 20, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 0, version 21
      Extended community: LB:3:192
      Origin-AS validity: not-found
```

次に、受信側（R2）ルータのDMZリンク帯域幅設定を表示する設定の例を示します。

```
RP/0/RP0/CPU0:router)# show bgp ipv4 unicast 10.1.1.1/32 detail

Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  1 3
    20.0.0.2 from 20.0.0.2 (10.0.0.81)
      Origin incomplete, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 0, version 17
      Extended community: LB:1:192
      Origin-AS validity: not-found
```

RPKI による BGP プレフィックスの発信元検証

BGP ルートは、BGP アナウンスメントの形で、プレフィックスが経由したドメイン間パスを識別する自律システム（AS）の設定と、アドレス プレフィックスを関連付けます。この設定は、BGP 内で AS_PATH 属性として表され、プレフィックスを発信した AS で開始されます。

誤ったプレフィックスのアナウンス、中間者攻撃など、BGPに対する既知の脅威を低減しやすくするためのセキュリティ要件の 1 つは、BGP ルートの発信元 AS を検証する能力です。アドレス プレフィックスの発信元であるとする AS 番号（BGP ルートの AS_PATH 属性から導出）は、プレフィックスの所有者によって検証および許可される必要があります。

Resource Public Key Infrastructure（RPKI）は、IP アドレスとリソースとしての AS 番号の公的で検証可能なデータベースを構築するためのアプローチです。RPKI は、BGP（インターネット）プレフィックスから許可された元の AS 番号への情報マッピングなどの情報を含む、グローバルに分散されたデータベースです。BGP を実行しているルータは、RPKI に接続して、BGP パスの元の AS を検証できます。

RPKI キャッシュ サーバの設定

リソース公開キーインフラストラクチャ（RPKI）キャッシュ サーバ パラメータを設定するには、次の作業を実行します。

RPKI サーバのコンフィギュレーション モードで RPKI キャッシュ サーバ パラメータを設定します。RPKI サーバ コンフィギュレーション モードを開始するには、ルータ BGP コンフィギュレーション モードで **rpki server** コマンドを使用します。

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 100
```

BGP AS 番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **rpki cache** {*host-name* | *ip-address*}

例：

```
RP/0/RP0/cpu 0: router(config-bgp)#rpki server 10.2.3.4
```

RPKI サーバのコンフィギュレーション モードを開始し、RPKI のキャッシュ パラメータを設定します。

ステップ 4 次のいずれかのコマンドを使用します。

- **transport ssh port** *port_number*
- **transport tcp port** *port_number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-rpki-server)#transport ssh port 22
```

または

```
RP/0/RP0/cpu 0: router(config-bgp-rpki-server)#transport tcp port 2
```

RPKI キャッシュの転送方法を指定します。

- **ssh**：SSH を使用して RPKI キャッシュに接続するには **ssh** を選択します。
- **tcp**：TCP（暗号化されていない）を使用して RPKI キャッシュに接続するには **tcp** を選択します。
- **port** *port_number*：指定した RPKI キャッシュ転送に使用するポート番号を指定します。TCP の場合、サポートされているポート番号の範囲は 1 ~ 65535 です。SSH の場合は、ポート番号 22 を使用します。

（注）
- SSH を介した RPKI キャッシュ転送の場合は、カスタム ポート番号を指定しないでください。SSH を介した RPKI にはポート 22 を使用する必要があります。
- transport には TCP と SSH のいずれかを設定できます。transport を変更すると、キャッシュ セッションがフラップします。

ステップ 5 （任意） **username** *user_name*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-rpki-server)#username ssh_rpki_cache
```

RPKI キャッシュ サーバの（SSH）ユーザ名を指定します。

ステップ 6 （任意） **password**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-rpki-server)#password ssh_rpki_pass
```

RPKI キャッシュ サーバの（SSH）パスワードを指定します。

（注） 「username」と「password」の設定は、SSH 転送方式がアクティブな場合にのみ適用されます。

ステップ 7 **preference** *preference_value*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-rpki-server)#preference 1
```

RPKI キャッシュのプリファレンス値を指定します。プリファレンス値の範囲は 1 ～ 10 です。設定するプリファレンス値は低い方が適切です。

ステップ 8 **purge-time** *time*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-rpki-server)#purge-time 30
```

キャッシュ セッションのドロップ後に、BGP がキャッシュからのルートを保持するまで待機する時間を設定します。破棄時間は秒単位で設定します。破棄時間の範囲は 30 ～ 360 秒です。

ステップ 9 次のいずれかのコマンドを使用します。

- **refresh-time** *time*
- **refresh-time off**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-rpki-server)#refresh-time 20
```

または

```
RP/0/RP0/cpu 0: router(config-bgp-rpki-server)#refresh-time off
```

キャッシュへの定期的なシリアルクエリー送信操作の間に BGP が待機する時間を設定します。リフレッシュの時間を秒単位で設定します。リフレッシュの時間の範囲は 15 ～ 3600 秒です。

シリアルクエリーを定期的に送信しないように指定するには、**off** オプションを設定します。

ステップ 10 次のいずれかのコマンドを使用します。

- **response-time** *time*
- **response-time off**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-rpki-server)#response-time 30
```

または

```
RP/0/RP0/cpu 0: router(config-bgp-rpki-server)#response-time off
```

シリアルまたはリセットのクエリーを送信した後に BGP が応答を待機する時間を設定します。応答時間を秒の単位で設定します。応答時間の範囲は 15 ～ 3600 秒です。

応答を無期限に待機するには、**off** オプションを設定します。

ステップ 11 **shutdown**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-rpki-server)#shutdown
```

RPKI キャッシュのシャット ダウンを設定します。

ステップ 12 **commit**

BGP プレフィックス検証の設定

リリース 6.5.1 以降、RPKI はデフォルトでディセーブルになっています。リリース 6.5.1 からは、次のタスクを使用して RPKI プレフィックス検証を設定します。

```
Router(config)# router bgp 100
/* The bgp origin-as validation time and bgp origin-as validity signal ibgp commands are
 optional. */.
Router(config-bgp)# bgp origin-as validation time 50
Router(config-bgp)# bgp origin-as validation time off
Router(config-bgp)# bgp origin-as validation signal ibgp
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# bgp origin-as validation enable
```

次のコマンドを使用して、origin-as 検証の設定を確認します。

```
Router# show bgp origin-as validity

Thu Mar 14 04:18:09.656 PDT
BGP router identifier 10.1.1.1, local AS number 1
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 514
BGP main routing table version 514
BGP NSR Initial initsync version 2 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
Origin-AS validation codes: V valid, I invalid, N not-found, D disabled
    Network                Next Hop            Metric LocPrf Weight Path
 *> 209.165.200.223/27     0.0.0.0                  0         32768 ?

 *> 209.165.200.225/27     0.0.0.0                  0         32768 ?

 *> 19.1.2.0/24            0.0.0.0                  0         32768 ?

 *> 19.1.3.0/24            0.0.0.0                  0         32768 ?

 *> 10.1.2.0/24            0.0.0.0                  0         32768 ?

 *> 10.1.3.0/24            0.0.0.0                  0         32768 ?

 *> 10.1.4.0/24            0.0.0.0                  0         32768 ?

 *> 198.51.100.1/24        0.0.0.0                  0         32768 ?

 *> 203.0.113.235/24       0.0.0.0                  0         32768 ?

V*> 209.165.201.0/27       10.1.2.1                 0         4002 i

N*> 198.51.100.2/24        10.1.2.1                 0         4002 i
```

```
I*> 198.51.100.1/24         10.1.2.1                  0          4002 i

 *> 192.0.2.1.0/24          0.0.0.0                    0         32768 ?
```

```
Router# show bgp process
Mon Jul  9 16:47:39.428 PDT

BGP Process Information:
...
Use origin-AS validity in bestpath decisions
Allow (origin-AS) INVALID paths
Signal origin-AS validity state to neighbors

Address family: IPv4 Unicast
...
Origin-AS validation is enabled for this address-family
Use origin-AS validity in bestpath decisions for this address-family
Allow (origin-AS) INVALID paths for this address-family
Signal origin-AS validity state to neighbors with this address-family
```

RPKI 最適パス計算の設定

RPKI 最適パス計算オプションを設定するには、次の作業を実行します。

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 100
```

BGP AS 番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **rpki bestpath use origin-as validity**

例：

```
RP/0/RP0/cpu 0: router(config-bgp)#rpki bestpath use origin-as validity
```

BGP 最適パス処理でのパスのプリファレンスに影響する BGP パスの有効性状態をイネーブルにします。この設定は、ルータ BGP アドレス ファミリ サブモードでも設定できます。

ステップ 4 **rpki bestpath origin-as allow invalid**

例：

```
RP/0/RP0/cpu 0: router(config-bgp)#rpki bestpath origin-as allow invalid
```

すべての「無効な」パスを BGP 最適パス計算対象にします。

（注） この設定はグローバル アドレス ファミリ、ネイバー、およびネイバー アドレス ファミリの各サブモードでも設定できます。ルータ BGP とアドレス ファミリ サブモードで rpki bestpath origin-as allow invalid を設定すると、すべての「無効な」パスが BGP 最適パス計算対象になります。デフォルトではこのようなパスは最適パス候補になりません。ネイバーまたはネイバー アドレス ファミリ サブモードで pki bestpath origin-as を設定すると、その特定のネイバーまたはネイバーアドレスファミリのすべての「無効な」パスが最適パス候補として見なされます。このネイバーは eBGP ネイバーでなければなりません。

この設定は、**rpki bestpath use origin-as validity** 設定がイネーブルの場合にのみ有効になります。

ステップ 5 commit

弾力性のある CE ごとのラベル割り当てモード

弾力性のある CE ごとのラベル割り当ては、CE ごとのラベル割り当てモードの拡張機能で、プレフィックス独立コンバージェンス（PIC）とロード バランシングをサポートします。現時点では、プレフィックス単位、CE ごと、および VRF ごとの 3 つのラベル割り当てモードに次の制限があります。

- ASR 9000 イーサネット ライン カードと A9K-SIP-700 に対するサポートなし
- PIC に対するサポートなし
- 複数の CE にわたるロード バランシングに対するサポートなし
- PIC をサポートするローカル トラフィックの迂回時の一時的な転送ループ
- EIBGP マルチパスのロード バランシングに対するサポートなし
- 転送パフォーマンスへの影響
- ネットワーク内の別のベンダールータでのプレフィックスごとのラベル割り当てモードによるスケールの問題

弾力性のある CE ごとのラベル割り当てスキームでは、CE パスまたはネクスト ホップのそれぞれの一意のセットに対して BGP が LSD に一意の書き換えラベルをインストールします。BGP テーブルにこのラベルをポイントする 1 つ以上のプレフィックスが含まれている場合があります。また、BGP は CE パス（プライマリ）と、オプションのバックアップ PE パスを RIB にインストールします。FIB は LSD からラベル書き換え情報を、RIB から IP パスを学習します。安定状態では、弾力性のある CE ごとのラベル宛のラベル付きのトラフィックには、すべての CE ネクスト ホップにわたってロード バランシングが行われます。すべての CE パスが失敗すると、そのラベル宛のすべてのトラフィックが IP ルックアップとなり、使用可能な場合は、バックアップ PE に転送されます。このアクションはラベルをポイントする可能性がある

プレフィックスの数と関係なく、ラベル上で実行されるため、プライマリ パスの障害時はPIC の動作になります。

VRFアドレスファミリでの弾力性のあるCEごとのラベル割り当てモードの設定

VRFアドレス ファミリに弾力性のあるCEごとのラベル割り当てモードを設定するには、このタスクを実行します。

手順

ステップ 1 **configure**

例：

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

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 **router bgp***as-number*

例：

```
RP/0/RP0/cpu 0: router(config)# router bgp 666
RP/0/RP0/cpu 0: router(config-bgp)#
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **vrf***vrf-instance*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# vrf vrf-pe
RP/0/RP0/cpu 0: router(config-bgp-vrf)#
```

VRF インスタンスを設定します。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

ステップ 5 **label-mode per-ce**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-vrf-af)# label-mode per-ce
RP/0/RP0/cpu 0: router(config-bgp-vrf-af)#
```

弾力性のある CE ごとのラベル割り当てモードを設定します。

ステップ 6 次のいずれかを実行します。

- **end**
- **commit**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-vrf-af)# end
```

または

```
RP/0/RP0/cpu 0: router(config-bgp-vrf-af)# 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。
  - **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。

- 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。

次に、VRF アドレス ファミリに弾力性のある CE ごとのラベル割り当てモードを設定する例を示します。

```
RP/0/RP0/cpu 0: router# configure
RP/0/RP0/cpu 0: router(config)# router bgp 666
RP/0/RP0/cpu 0: router(config-bgp)# vrf vrf-pe
RP/0/RP0/cpu 0: router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/RP0/cpu 0: router(config-bgp-vrf-af)# label-mode per-ce
RP/0/RP0/cpu 0: router(config-bgp-vrf-af)# end
```

ルートポリシーを使用した弾力性のあるCEごとのラベル割り当てモードの設定

ルート ポリシーを使用して弾力性のある CE ごとのラベル割り当て モードを設定するには、このタスクを実行します。

手順

ステップ 1 **configure**

例：

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

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** *policy-name*

例：

```
RP/0/RP0/cpu 0: router(config)# route-policy route1
RP/0/RP0/cpu 0: router(config-rpl)#
```

ルート ポリシーを作成し、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。

ステップ 3 **set label-mode per-ce**

例：

```
RP/0/RP0/cpu 0: router(config-rpl)# set label-mode per-ce
RP/0/RP0/cpu 0: router(config-rpl)#
```

弾力性のある CE ごとのラベル割り当てモードを設定します。

ステップ 4 次のいずれかを実行します。

- **end**
- **commit**

例：

```
RP/0/RP0/cpu 0: router(config-rpl)# end
```

または

```
RP/0/RP0/cpu 0: router(config-rpl)# 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。
- **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。

- 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。

次に、ルート ポリシーを使用して弾力性のある CE ごとのラベル割り当てモードを設定する例を示します。

```
RP/0/RP0/cpu 0: router# configure
RP/0/RP0/cpu 0: router(config)# route-policy route1
RP/0/RP0/cpu 0: router(config-rpl)# set label-mode per-ce
RP/0/RP0/cpu 0: router(config-rpl)# end
```

BGP VRF ダイナミック ルートのリーク

Border Gateway Protocol（BGP）ダイナミック ルートのリーク機能では、デフォルトの VRF（グローバル VRF）とその他すべての非デフォルト VRF 間にルートをインポートできるようにし、グローバルと VPN ホスト間に接続を提供します。インポート プロセスによって VRF テーブルにインターネット ルートが組み込まれるか、またはインターネット テーブルに VRF ルートが組み込まれて、接続を提供します。

（注） 直接接続されたルートは、デフォルトの VRF から非デフォルトの VRF に BGP VRF ダイナミック ルート リークを使用してリークできません。

ダイナミック ルート リークは次の方法で有効になります。

- VRF アドレスファミリ コンフィギュレーション モードで **import from default-vrf route-policy** *route-policy-name*［**advertise-as-vpn**］コマンドを使用して、デフォルト VRF から非デフォルト VRF にインポートする。

  advertise-as-vpn オプションが設定されている場合、デフォルト VRF から非デフォルト VRF にインポートしたパスは、PE と CE にアドバタイズされます。**advertise-as-vpn** オプションが設定されていない場合、デフォルト VRF から非デフォルト VRF にインポートされたパスは PE にアドバタイズされません。ただし、この場合も CE にはパスがアドバタイズされます。

- VRF アドレスファミリ コンフィギュレーション モードで **export to default-vrf route-policy** *route-policy-name* コマンドを使用して、非デフォルト VRF からデフォルト VRF にインポートする。

インポートしたルートをフィルタリングするには、ルート ポリシーが必要です。これにより、インターネット テーブルと VRF テーブル間でのルートの意図せぬインポートや対応するセキュリティ問題を低減します。インポートできるプレフィックスの数にハードリミットはありません。インポートによりインポート先の VRF に新しいプレフィックスが作成されるため、プレフィックスとパスの総数が増加します。ただし、グルーバル ルートをインポートしている各 VRF がグローバル テーブルを受け取るネイバーと同等のワークロードを追加します。これは、ユーザが一部を除くすべてのプレフィックスをフィルタリングした場合も同様です。したがって、インポートする VRF の適切な数は 5 ～ 10 個です。

VRF ダイナミック ルートのリークの設定

次のステップを実行して、デフォルト VRF から非デフォルト VRF にルートをインポートするか、または非デフォルト VRF からデフォルト VRF にルートをインポートします。

始める前に

ダイナミック ルート リークを設定するには、ルート ポリシーが必要です。ルート ポリシーを設定するには、グローバル コンフィギュレーション モードで**route-policy** *route-policy-name* コマンドを使用します。

手順

ステップ 1 **configure**

ステップ 2 **vrf** *vrf_name*

例：

```
RP/0/RSP0/CPU0:PE51_ASR-9010(config)#vrf vrf_1
```

VRF コンフィギュレーション モードを開始します。

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

例：

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

VRF アドレス ファミリ コンフィギュレーション モードを開始します。

ステップ 4 次のいずれかのオプションを使用します。

- **import from default-vrf route-policy** *route-policy-name* [**advertise-as-vpn**]
- **export to default-vrf route-policy** *route-policy-name*

例：

```
RP/0/RP0/cpu 0: router(config-vrf-af)#import from default-vrf route-policy
rpl_dynamic_route_import
```

または

```
RP/0/RP0/cpu 0: router(config-vrf-af)#export to default-vrf route-policy
rpl_dynamic_route_export
```

デフォルト VRF から非デフォルト VRF にルートをインポートするか、または非デフォルト VRF からデフォルト VRF にルートをインポートします。

- **import from default-vrf**：デフォルト VRF から非デフォルト VRF へのインポートを設定します。

  advertise-as-vpn オプションが設定されている場合、デフォルト VRF から非デフォルト VRF にインポートしたパスは、PE と CE にアドバタイズされます。**advertise-as-vpn** オプションが設定されていない場合、デフォルト VRF から非デフォルト VRF にインポートされたパスは PE にアドバタイズされません。ただし、この場合も CE にはパスがアドバタイズされます。

- **export to default-vrf**：非デフォルト VRF からデフォルト VRF へのインポートを設定します。デフォルト VRF からインポートされたパスが他の PE にアドバタイズされます。

ステップ 5 **commit**

VRF ダイナミック ルートの設定：例

デフォルト VRF から非デフォルト VRF へのルートのインポート：

```
vrf vrf_1
 address-family ipv6 unicast
  import from default-vrf route-policy rpl_dynamic_route_import
 !
end
```

非デフォルト VRF からデフォルト VRF へのルートのインポート：

```
vrf vrf_1
 address-family ipv6 unicast
    export to default-vrf route-policy rpl_dynamic_route_export
 !
end
```

次のタスク

次の **show bgp** コマンドの出力には、ダイナミック ルート リーク設定の情報が表示されます。

- **show bgp prefix** コマンドを使用すると、インポートしたパスの送信元 RD と送信元 VRF が表示されます。これには、IPv4 または IPv6 ユニキャスト プレフィックスにインポートしたパスがある場合も含まれます。

- **show bgp imported-routes** コマンドを使用すると、デフォルト VRF の IPv4 ユニキャストおよび IPv6 ユニキャストのアドレスファミリが表示されます。

BGP での VPN ルーティングおよび転送インスタンスの設定

機能を設定するライン カード スロットに使用可能なレイヤ 3 VPN ライセンスがある場合に限り、レイヤ 3（仮想プライベート ネットワーク）を設定できます。拡張 IP ライセンスが有効になっている場合、インターフェイスで 4096 レイヤ 3 VPN ルーティングおよび転送インスタンス（VRF）を設定できます。インフラストラクチャ VRF のライセンスが有効な場合は、8 つのレイヤ 3 VRF をラインカードに設定できます。

適切なライセンスが有効になっていないと、次のエラー メッセージが表示されます。

```
RP/0/RP0/cpu 0: router#LC/0/0/CPU0:Dec 15 17:57:53.653 : rsi_agent[247]:
%LICENSE-ASR9K_LICENSE-2-INFRA_VRF_NEEDED : 5 VRF(s) are configured without license
A9K-iVRF-LIC in violation of the Software Right To Use Agreement.
This feature may be disabled by the system without the appropriate license.
Contact Cisco to purchase the license immediately to avoid potential service interruption.
```

（注） L2VPN サービスの設定に AIP ライセンスは必要ありません。

次の作業は、BGP に VPN ルーティングおよび転送（VRF）インスタンスを設定する場合に実行します。

プロバイダー エッジ ルータでの仮想ルーティングおよび転送テーブルの定義

プロバイダー エッジ（PE）ルータに VPN ルーティングおよび転送（VRF）テーブルを定義するには、次の作業を実行します。

手順

ステップ 1 **configure**

ステップ 2 **vrf** *vrf-name*

例：

```
RP/0/RP0/cpu 0: router(config)# vrf vrf_pe
```

VRF インスタンスを設定します。

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

例：

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

IPv4 または IPv6 のいずれかのアドレス ファミリを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。

このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 4 **maximum prefix** *maximum* [*threshold*]

例：

```
RP/0/RP0/cpu 0: router(config-vrf-af)# maximum prefix 2300
```

VRF テーブルで許可するプレフィックスの数に制限を設定します。

ルートの最大数はダイナミック ルーティング プロトコルと、スタティックまたは接続されたルートに適用されます。

mid-threshold 引数を使用して、プレフィックスを制限するしきい値のパーセンテージを指定できます。

ステップ 5 **import route-policy** *policy-name*

例：

```
RP/0/RP0/cpu 0: router(config-vrf-af)# import route-policy policy_a
```

（任意）VRF にインポートする内容をより細かく制御します。このインポート フィルタでは、指定された *policy-name* 引数に一致しないプレフィックスは破棄されます。

ステップ 6 **import route-target** [*as-number* **:** *nn* | *ip-address* **:** *nn*]

例：

```
RP/0/RP0/cpu 0: router(config-vrf-af)# import route-target 234:222
```

ルート ターゲット（RT）拡張コミュニティのリストを指定します。指定されたインポート ルート ターゲット拡張コミュニティと関連付けられているプレフィックスだけが VRF にインポートされます。

ステップ 7 **export route-policy** *policy-name*

例：

```
RP/0/RP0/cpu 0: router(config-vrf-af)# export route-policy policy_b
```

（任意）VRF にエクスポートする内容をより細かく制御します。このエクスポート フィルタでは、指定された *policy-name* 引数に一致しないプレフィックスは破棄されます。

ステップ 8 **export route-target** [*as-number* **:** *nn* | *ip-address* **:** *nn*]

例：

```
RP/0/RP0/cpu 0: routerr(config-vrf-af)# export route-target 123;234
```

ルート ターゲット拡張コミュニティのリストを指定します。エクスポート ルート ターゲット コミュニティは、リモート PE にアドバタイズされる際にプレフィックスと関連付けられます。

リモート PE は、これらのエクスポート ルート ターゲット コミュニティと一致するインポート RT を持つ VRF に、これらのプレフィックスをインポートします。

ステップ 9 **commit**

ルート識別子の設定

ルート識別子（RD）により、複数の VPN ルーティングおよび転送（VRF）インスタンスにおいてプレフィックスが固有になります。

L3VPN マルチパス同一ルート識別子（RD）環境では、プレフィックスを RIB にインストールするかどうかは、プレフィックスの最適パスに基づいて決まります。稀に設定が誤っている場合（最適パスが RIB にインストールできる有効なパスではない場合）、BGP はプレフィックスをドロップし、その他のパスを考慮しません。この動作は RD のセットアップによって異なります。最適マルチパスが RIB にインストールするパスとして無効な場合には、非最適マルチパスがインストールされます。

RD を設定するには、次の作業を実行します。

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 120
```

BGP コンフィギュレーション モードを開始します。このモードでは BGP ルーティング プロセスを設定できます。

ステップ 3 **bgp router-id** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# bgp router-id 10.0.0.0
```

BGP スピーキング ルータの固定ルータ ID を設定します。

ステップ 4 **vrf** *vrf-name*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# vrf vrf_pe
```

VRF インスタンスを設定します。

ステップ 5 **rd** { *as-number* **:** *nn* | *ip-address* **:** *nn* | **auto** }

例：

```
RP/0/RP0/cpu 0: router(config-bgp-vrf)# rd 345:567
```

ルート識別子を設定します。

ルータが自動的に一意の RD を VRF に割り当てるようにする場合は、 **auto** キーワードを使用します。

ルータ コンフィギュレーション モードで **bgp router-id** コマンドを使用してルータ ID が設定されている場合にのみ、RD を自動で割り当てることができます。これにより、自動 RD 生成に使用できるグローバルで固有のルータ ID を設定できます。VRF のルータ ID はグローバルで固有である必要はありません。また、自動 RD 生成で VRF ルータ ID を使用することは正しくありません。ルータ ID を 1 つにすると、いつ再起動してもルータ ID が固定であるため、BGP グレースフル リスタートで RD 情報のチェックポイントも行いやすくなります。

ステップ 6 次のいずれかを実行します。

- **end**
- **commit**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-vrf)# end
```

または

```
RP/0/RP0/cpu 0: router(config-bgp-vrf)# 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** を入力すると、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 XR EXEC モードに戻ります。
  - **no** を入力すると、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 XR 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。

- 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。

PE-PE または PE-RR 内部 BGP セッションの設定

BGP がプロバイダー エッジ（PE）ルータ間で VPN 到着達可能性情報を送信できるようにするには、PE-PE 内部 BGP（iBGP）セッションを設定する必要があります。PE はリモート PE ルー

タから送信される VPN 情報を使用して VPN 接続と使用するラベル値を判別します。これにより、リモート（出力）ルータはパケット転送で正しい VPN へのパケットを逆多重化できます。

PE ルータで設定されている VPN に接続するすべての PE および RR ルータに対して PE-PE、PE ルート リフレクタ（RR）iBGP セッションが定義されます。

PE-PE iBGP セッションを設定し、PE でグローバル VPN オプションを設定するには、次の作業を実行します。

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 120
```

自律システム番号を指定し、BGP コンフィギュレーション モードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。

ステップ 3 **address-family vpnv4 unicast**

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# address-family vpvn4 unicast
```

VPN アドレス ファミリ コンフィギュレーション モードを開始します。

ステップ 4 **exit**

例：

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

現在のコンフィギュレーション モードを終了します。

ステップ 5 **neighbor** *ip-address*

例：

```
RP/0/RP0/cpu 0: router(config-bgp)# neighbor 172.16.1.1
```

PE の iBGP ネイバーを設定します。

ステップ 6 **remote-as** *as-number*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# remote-as 1
```

ネイバーをリモート自律システム番号に割り当てます。

ステップ 7 **description** *text*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# description neighbor 172.16.1.1
```

（任意）ネイバーの説明を指定します。description は、コメントを保存するために使用されます。ソフトウェアの機能には影響しません。

ステップ 8 **password** { **clear** | **encrypted** } *password*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# password encrypted 123abc
```

2 つの BGP ネイバーの間の TCP 接続上で Message Digest 5（MD5）認証をイネーブルにします。

ステップ 9 **shutdown**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# shutdown
```

指定されたネイバーのあらゆるアクティブセッションを終了し、すべての関連するルーティング情報を削除します。

ステップ 10 **timers** *keepalive hold-time*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# timers 12000 200
```

BGP ネイバーのタイマーを設定します。

ステップ 11 **update-source** *type interface-id*

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# update-source gigabitEthernet 0/1/5/0
```

ネイバーとのiBGPセッションを形成するときに、iBGPセッションが特定のインターフェイスのプライマリ IP アドレスをローカル アドレスとして使用できるようにします。

ステップ 12 **address-family vpnv4 unicast**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr)# address-family vpvn4 unicast
```

VPN ネイバー アドレス ファミリ設定モードを開始します。

ステップ 13 **route-policy** *route-policy-name* **in**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# route-policy pe-pe-vpn-in in
```

着信ルートのルーティングポリシーを指定します。ポリシーを使用すると、ルートのフィルタリングやルート属性の変更ができます。

ステップ 14 **route-policy** *route-policy-name* **out**

例：

```
RP/0/RP0/cpu 0: router(config-bgp-nbr-af)# route-policy pe-pe-vpn-out out
```

発信ルートのルーティングポリシーを指定します。ポリシーを使用すると、ルートのフィルタリングやルート属性の変更ができます。

ステップ 15 **commit**

PE-CE プロトコルとしての BGP の設定

PE で BGP を設定し、BGP を使用した PE-CE 通信を確立するには、次の作業を実行します。

手順

	コマンドまたはアクション	目的
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 120`	自律システム番号を指定し、BGP コンフィギュレーションモードを開始します。このモードでは、BGP ルーティング プロセスを設定できます。
ステップ 3	**vrf** *vrf-name* 例： `RP/0/RP0/cpu 0: router(config-bgp)# vrf vrf_pe_2`	PE ルータで特定の VRF の BGP ルーティングをイネーブルにします。
ステップ 4	**bgp router-id** *ip-address* 例： `RP/0/RP0/cpu 0: router(config-bgp-vrf)# bgp router-id 172.16.9.9`	BGP スピーキング ルータの固定ルータ ID を設定します。
ステップ 5	**label-allocation-mode per-ce** 例： `RP/0/RP0/cpu 0: router(config-bgp-vrf)# label-allocation-mode per-ce`	• **per-ce** キーワードは、CE ごとのラベル割り当てモードを設定して PE ルータでの追加ルックアップを回避し、ラベル スペースを節約します（デフォルトのラベル割り当てモードはプレフィックス単位で

	コマンドまたはアクション	目的
		す）。このモードでは、PEルータは、すべての即時ネクストホップ（ほとんどの場合、これはCEルータ）に 1 個のラベルを割り当てます。このラベルはネクストホップに直接マップされるため、データ転送中に VRF ルート ルックアップが実行されることはありません。ただし、割り当てられるラベルの数は、各 VRF に 1 つではなく、各 CE に 1 個です。BGP はすべてのネクストホップを認識するため、各ネクストホップにラベルを割り当てます（各 PE-CE インターフェイスではありません）。発信インターフェイスがマルチアクセスインターフェイスで、ネイバーのメディア アクセス コントロール（MAC）アドレスが不明な場合は、アドレス解決プロトコル（ARP）がパケット転送の間にトリガーされます。 • **per-vrf** キーワードは、一意の VRF からアドバタイズされたすべてのルートに同じラベルを使用するように設定します。
ステップ 6	**address-family** { **ipv4** | **ipv6** } **unicast** 例： `RP/0/RP0/cpu 0: router(config-vrf)# address-family ipv4 unicast`	IPv4 または IPv6 のいずれかのアドレス ファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。 このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。
ステップ 7	**network** { *ip-address* / *prefix-length* | *ip-address mask* } 例： `RP/0/RP0/cpu 0: router(config-bgp-vrf-af)# network 172.16.5.5`	VRF のコンテキストでアドレス ファミリのテーブルのネットワーク プレフィックスを発信します。

	コマンドまたはアクション	目的

ステップ 8	**aggregate-address** *address* / *mask-length*
	例：
	`RP/0/RP0/cpu 0:`
	`router(config-bgp-vrf-af)#`
	`aggregate-address 10.0.0.0/24`
		コアに保持されている状態を削減するためルーティング情報を集約するように VRF アドレス ファミリ コンテキストで集約を設定します。この集約により、PEエッジでの効率がいくらか低下します。これはパケットの最終ネクスト ホップを決定するために、さらにルックアップが必要になるためです。設定すると、一連のコンポーネントプレフィックスの代わりにサマリープレフィックスがアドバタイズされます。これはより詳細な集約です。PEは集約のラベルを 1 つだけアドバタイズします。コンポーネントプレフィックスでは CE へのネクスト ホップが異なることがあるため、データ転送時に追加のルックアップを実行する必要があります。

ステップ 9	**exit**
	例：
	`RP/0/RP0/cpu 0:`
	`router(config-bgp-vrf-af)# exit`
		現在のコンフィギュレーションモードを終了します。

ステップ 10	**neighbor** *ip-address*
	例：
	`RP/0/RP0/cpu 0:`
	`router(config-bgp-vrf)# neighbor 10.0.0.0`
		CE ネイバーを設定します。 *ip-address* 引数は、プライベートアドレスにする必要があります。

ステップ 11	**remote-as** *as-number*
	例：
	`RP/0/RP0/cpu 0:`
	`router(config-bgp-vrf-nbr)# remote-as 2`
		CE ネイバーのリモート AS を設定します。

ステップ 12	**password** { **clear** | **encrypted** } *password*
	例：
	`RP/0/RP0/cpu 0:`
	`router(config-bgp-vrf-nbr)# password encrypted 234xyz`
		2 つの BGP ネイバー間の TCP 接続で Message Digest 5（MD5）認証をイネーブルにします。

	コマンドまたはアクション	目的

ステップ 13	**ebgp-multihop** [*ttl-value*]
	例：
	`RP/0/RP0/cpu 0:`
	`router(config-bgp-vrf-nbr)#`
	`ebgp-multihop 55`
		直接接続していないネットワーク上の外部ピアへの BGP 接続を受け入れて試行するように CE ネイバーを設定します。

ステップ 14	次のいずれかを実行します。
	• **address-family** { **ipv4** | **ipv6** } **unicast**
	• **address-family {ipv4 {unicast | labeled-unicast} | ipv6 unicast}**
	例：
	`RP/0/RP0/cpu 0: router(config-vrf)#`
	`address-family ipv4 unicast`
		IPv4 または IPv6 のいずれかのアドレスファミリ ユニキャストを指定し、アドレス ファミリのコンフィギュレーション サブモードを開始します。
		このコマンドのすべてのキーワードと引数のリストを参照するには、CLI ヘルプ（?）を使用します。

ステップ 15	**site-of-origin** [*as-number* **:** *nn* | *ip-address* **:** *nn*]
	例：
	`RP/0/RP0/cpu 0:`
	`router(config-bgp-vrf-nbr-af)#`
	`site-of-origin 234:111`
		site-of-origin（SoO）拡張コミュニティを設定します。この CE ネイバーから学習されたルートは、その他の PE にアドバタイズされる前に SoO 拡張コミュニティのタグが付けられます。PE ルータで as-override が設定されている場合にループを検出する目的で SoO が使用されることがよくあります。プレフィックスが同じサイトにループする場合、PE はこのことを検出して CE に更新を送信しません。

ステップ 16	**as-override**
	例：
	`RP/0/RP0/cpu 0:`
	`router(config-bgp-vrf-nbr-af)#`
	`as-override`
		PE ルータで AS オーバーライドを設定します。これにより PE ルータは CE の ASN をそれ自体の（PE）ASN に置き換えます。
		（注） この情報が失われることが原因でルーティング ループが発生することがあります。as-override によって引き起こされるループを防ぐには、as-override と site-of-origin を組み合わせて使用します。

	コマンドまたはアクション	目的

ステップ 17	**allowas-in** [*as-occurrence-number*]
	例：
	`RP/0/RP0/cpu 0:`
	`router(config-bgp-vrf-nbr-af)#`
	`allowas-in 5`
		PE 自律システム番号（ASN）を持つ AS パスを指定された回数だけ許可します。
		ハブ アンド スポーク型 VPN ネットワークは、HUB CE を通じて、HUB PE へのルーティング情報のループバックを必要とします。この場合、PE ASN が存在するために HUB PE によってループバック情報がドロップされます。これを回避するため、PE ASN が指定された回数に達していても **allowas-in** コマンドを使用してプレフィックスを許可します。

ステップ 18	**route-policy** *route-policy-name* **in**
	例：
	`RP/0/RP0/cpu 0:`
	`router(config-bgp-vrf-nbr-af)#`
	`route-policy pe_ce_in_policy in`
		着信ルートのルーティング ポリシーを指定します。ポリシーを使用すると、ルートのフィルタリングやルート属性の変更ができます。

ステップ 19	**route-policy** *route-policy-name* **out**
	例：
	`RP/0/RP0/cpu 0:`
	`router(config-bgp-vrf-nbr-af)#`
	`route-policy pe_ce_out_policy out`
		発信ルートのルーティング ポリシーを指定します。ポリシーを使用すると、ルートのフィルタリングやルート属性の変更ができます。

ステップ 20	**commit**	

リンク障害後の eBGP セッションの即時リセット

デフォルトでは、リンクがダウンすると、直接隣接する外部ピアの BGP セッションはすべて即時にリセットされます。自動リセットをディセーブルにするには **bgp fast-external-fallover disable** コマンドを使用します。自動リセットをイネーブルにするには **no bgp fast-external-fallover disable** コマンドを使用します。

BGP タイマー値が 10 および 30 に設定されているノードで eBGP セッションの数が 3500 に達すると、eBGP セッションはフラップします。3500 を超える数の eBGP セッションに対応するには、**lpts pifib hardware police location** *location-id* コマンドを使用してパケット レートを大きくします。eBGP セッションを増加する設定の例を次に示します。

```
RP/0/RP0/cpu 0: router#configure
RP/0/RP0/cpu 0: router(config)#lpts pifib hardware police location 0/2/CPU0
RP/0/RP0/cpu 0: router(config-pifib-policer-per-node)#flow bgp configured rate 4000
RP/0/RP0/cpu 0: router(config-pifib-policer-per-node)#flow bgp known rate 4000
```

```
RP/0/RP0/cpu 0: router(config-pifib-policer-per-node)#flow bgp default rate 4000
RP/0/RP0/cpu 0: router(config-pifib-policer-per-node)#commit
```

BGP の実装に関する概要

BGP を実装するには、次の概念を理解する必要があります。

BGP ルータ ID

ネイバー間に BGP セッションを確立するには、BGP にルータ ID を割り当てる必要があります。ルータ ID は、BGP セッションが確立されると、OPEN メッセージに含めて BGP ピアに送信されます。

BGP は次の方法（プリファレンス順）でルータ ID の取得を試みます。

- ルータ コンフィギュレーション モードで **bgp router-id** コマンドを使用して設定されたアドレスを使用する。
- 保存されたループバック アドレス設定を使用してルータがブートされた場合に、システムのループバック インターフェイス上の最大の IPv4 アドレスを使用する。
- 保存された設定に存在しない場合に、設定される最初のループバック アドレスのプライマリ IPv4 アドレスを使用する。

このいずれの方法でもルータ ID を取得できない場合、BGP はルータ ID を持たず、BGP ネイバーとのピアリング セッションを確立できません。そのような場合は、エラー メッセージがシステム ログに記録され、**show bgp summary** コマンドでは、ルータ ID として 0.0.0.0 が表示されます。ルータ ID を取得した BGP では、さらに適したルータ ID が使用可能になっても、同じルータ ID の使用を続行します。この使用方法によって、いずれの BGP セッションでも不要なフラッピングが発生しないようにします。一方、現在使用中のルータ ID が無効になった場合（インターフェイスがダウンするか、設定が変更されたことによる）、BGP では新しいルータ ID を選択し（上記のルールを使用）、確立したすべてのピアリング セッションをリセットします。

（注） ルータ ID の不要な変更（およびそれによる BGP セッションのフラッピング）を避けるために、**bgp router-id** コマンドを設定することを、強く推奨します。

BGP のデフォルト制限

BGP では、ルータに設定できるネイバーの最大数、および特定のアドレス ファミリのピアから受け入れるプレフィックスの最大数に制限を設定しています。この制限は、ルータにとって、ローカルまたはリモート ネイバーのいずれかの設定ミスに起因する、リソースの枯渇に対する予防措置となります。BGP 設定には、次の制限が適用されます。

- 設定できるピアのデフォルトの最大数は 4000 です。このデフォルトは、**bgp maximum neighbor** コマンドを使用して変更できます。制限の範囲は 1 ～ 15000 です。 最大制限値を超えてさらにピアを設定しようとしたり、現在設定されているピアの数未満の最大制限値を設定しようとしたりすると失敗します。
- アドバタイズメントによりピアがBGPをフラッディングしないようにするために、サポートされているアドレス ファミリごとに、1つのピアから受け入れるプレフィックスの数に対する制限が課されます。デフォルトの制限値は、該当するアドレスファミリのピアに対して maximum-prefix *limit* コマンドを設定することにより、上書きできます。ユーザがそのアドレス ファミリに対するプレフィックスの最大数を設定していない場合は、次のデフォルト制限値が使用されます。
  - IPv4 ユニキャストに対する 512K（524,288）のプレフィックス
  - IPv6 ユニキャストに対する 128K（131,072）のプレフィックス
  - VPNv4 ユニキャストに対する 512K（524,288）のプレフィックス

  特定のアドレス ファミリのピアから受信したプレフィックスの数が、このアドレス ファミリに対する最大制限値（デフォルト設定またはユーザ設定のいずれかによる）を超えると、停止通知メッセージがそのネイバーに送信され、このネイバーとのピアリングが終了されます。

  特定のアドレスファミリのネイバーとのピアリングが確立され、そのネイバーから一定数のプレフィックスをすでに受信した後で、そのネイバーのプレフィックスの最大数が設定されていることがあります。設定されたプレフィックスの最大数が、アドレスファミリのネイバーからすでに受信したプレフィックスの数よりも小さい場合は、設定直後に停止通知メッセージがそのネイバーに送信され、そのネイバーとのピアリングが終了されます。

BGP 属性と演算子

このテーブルでは、接続点ごとの BGP 属性と演算子をまとめます。

表 *2 : BGP* 属性と演算子

接続点	属性	一致	セット
aggregation	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	community	is-empty matches-any matches-every	set set additive delete in delete not in delete all
	destination	in	—
	extcommunity cost	—	set set additive
	local-preference	is、ge、le、eq	set
	med	is、eg、ge、le	setset +set -
	next-hop	in	set
	origin	is	set
	source	in	—
	suppress-route	—	suppress-route
	weight	—	set

接続点	属性	一致	セット
allocate-label	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	community	is-empty matches-any matches-every	—
	destination	in	—
	label	—	set
	local-preference	is、ge、le、eq	—
	med	is、eg、ge、le	—
	next-hop	in	—
	origin	is	—
	source	in	—
clear-policy	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—

接続点	属性	一致	セット
dampening	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	community	is-empty matches-any matches-every	—
	dampening	—/	set dampening
	destination	in	—
	local-preference	is、ge、le、eq	—
	med	is、eg、ge、le	—
	next-hop	in	—
	origin	is	—
	source	in	—
debug	destination	in	—
default originate	med	—	set set + set -
	rib-has-route	in	—

接続点	属性	一致	セット
neighbor-in	as-path	in is-local length NA neighbor-is originates-from passes-through unique-length	prepend prepend most-recent remove as-path private-as replace
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	communitycommunity with ‘peeras’	is-empty matches-any matches-every	set set additive delete-in delete-not-in delete-all
	destination	in	—
	extcommunity cost	—	set set additive
	extcommunity rt	is-empty matches-any matches-every matches-within	set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every matches-within	—
	local-preference	is、ge、le、eq	set
	med	is、eg、ge、le	set set + set -

接続点	属性	一致	セット
	next-hop	in	set set peer address
	origin	is	set
	route-aggregated	route-aggregated	NA
	source	in	—
	weight	—	set

接続点	属性	一致	セット
neighbor-out	as-path	in is-local length — neighbor-is originates-from passes-through unique-length	prepend prepend most-recent remove as-path private-as replace
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	communitycommunity with ‘peeras’	is-empty matches-any matches-every	set set additive delete-in delete-not-in delete-all
	destination	in	—
	extcommunity cost	—	set set additive
	extcommunity rt	is-empty matches-any matches-every matches-within	set additive delete-in delete-not-in delete-all
	extcommunity soo	is-empty matches-any matches-every matches-within	—
	local-preference	is、ge、le、eq	set
	med	is、eg、ge、le	

接続点	属性	一致	セット
			set set + set - set max-unreachable set igp-cost
	next-hop	in	set set self
	origin	is	set
	path-type	is	—
	rd	in	—
	route-aggregated	route-aggregated	—
	source	in	—
	unsuppress-route	—	unsuppress-route
	vpn-distinguisher	—	set
neighbor-orf	orf-prefix	in	n/a

接続点	属性	一致	セット
network	as-path	—	prepend
	community	—	set set additive delete-in delete-not-in delete-all
	destination	in	—
	extcommunity cost	—	set set additive
	mpls-label	route-has-label	—
	local-preference	—	set
	med	—	set set+ set-
	next-hop	in	set
	origin	—	set
	route-type	is	—
	tag	is、ge、le、eq	—
	weight	—	set
next-hop	destination	in	—
	protocol	is、in	—
	source	in	—

接続点	属性	一致	セット
redistribute	as-path	—	prepend
	community	—	set set additive delete in delete not in delete all
	destination	in	—
	extcommunity cost	—	setset additive
	local-preference	—	set
	med	—	set set+ set-
	next-hop	in	set
	origin	—	set
	mpls-label	route-has-label	—
	route-type	is	—
	tag	is、eq、ge、le	—
	weight	—	set
retain-rt	extcommunity rt	is-empty matches-any matches-every matches-within	—

接続点	属性	一致	セット
show	as-path	in is-local length neighbor-is originates-from passes-through unique-length	—
	as-path-length	is、ge、le、eq	—
	as-path-unique-length	is、ge、le、eq	—
	community	is-empty matches-any matches-every	—
	destination	in	—
	extcommunity rt	is-empty matches-any matches-every matches-within	—
	extcommunity soo	is-empty matches-any matches-every matches-within	—
	med	is、eg、ge、le	—
	next-hop	in	—
	origin	is	—
	source	in	—

一部の BGP ルート属性は、さまざまな理由のため一部の BGP 接続点からアクセスできません。たとえば、set med igp-cost only コマンドは、設定された IGP コストがあり、ソース値を示す場合に意味があります。

次の表では、どの操作が有効であるか、およびどの場合に有効であるかをまとめます。

表 *3*：接続点により制限された *BGP* 動作

コマンド	インポート	エクスポート	集約	再配布
prepend as-path most-recent	eBGP のみ	eBGP のみ	該当なし	該当なし
replace as-path	eBGP のみ	eBGP のみ	該当なし	該当なし
set med igp-cost	禁止	eBGP のみ	禁止	禁止
set weight	該当なし	禁止	該当なし	該当なし
suppress	禁止	禁止	該当なし	禁止

BGP 最適パス アルゴリズム

BGP ルータは、通常は同じ宛先に対する複数のパスを受信します。BGP の最適パス アルゴリズムは、IP ルーティング テーブルに格納し、トラフィックの転送に使用する最適なパスを決めるものです。この項では、インターネット技術特別調査委員会（IETF）のネットワーク ワーキング グループによる draft-ietf-idr-bgp4-24.txt 資料の 9.1 項で指定されている BGP 最適パス アルゴリズムの Cisco IOS XR ソフトウェア実装について説明します。

BGP 最適パス アルゴリズムは、次の 3 つのパートに分かれて実行されます。

- パート 1：2 つのパスを比較して、いずれが優れているのかを判別します。
- パート 2：すべてのパスを順に処理し、全体として最適なパスを選択するためにパスを比較する順序を決定します。
- パート 3：新しい最適パスを使用するに足るだけの差が新旧の最適パスにあるかどうかを判別します。

（注） 比較演算が推移的ではないため、パート 2 で決定された比較の順序は重要です。つまり、3 つのパス、A、B、C がある場合、A と B を比較したときに A の方が優れていて、B と C と比較したときに B の方が優れている場合、A と C を比較したときに必ずしも A が優れているとは限りません。この非推移性は、Multi Exit Discriminator（MED）が、すべてのパス間ではなく、同じネイバー自律システム（AS）からのパス間のみで比較されるために生じます。

パスのペアの比較

2 つのパスを比較して、優れたパスを判別するには、次の手順を実行します。

1. いずれかのパスが無効な場合（可能な最大MED値を持つパス、到達不能なネクストホップを持つパスなど）、もう一方のパスが選択されます（そのパスが有効な場合）。
2. パスの準最適パス コスト コミュニティが等しくない場合は、準最適パス コスト コミュニティの低いパスが最適パスとして選択されます。
3. パスの重みが等しくない場合は、重みが最大のパスが選択されます。

（注） 重みは完全にルータにローカルであり、weightコマンドまたはルーティング ポリシーを使用して設定できます。

4. パスのローカルプリファレンスが等しくない場合は、ローカルプリファレンスが高い方のパスが選択されます。

（注） パスとともにローカル プリファレンス属性を受信したか、ルーティング ポリシーによって設定された場合は、その値が、この比較で使用されます。それ以外の場合は、デフォルト ローカル プリファレンス値の 100 が使用されます。デフォルト値は、bgp default local-preference コマンドを使用して変更できます。

5. パスの 1 つが再配布されたパス、つまり **redistribute** コマンドまたは **network** コマンドによるパスの場合は、そのパスが選択されます。それ以外の場合、パスの 1 つがローカルで作成された集約パスのとき、つまり aggregate-address コマンドによるパスのときは、そのパスが選択されます。

（注） ステップ 1 ～ ステップ 4 では、RFC 1268 の「Path Selection with BGP」を実装します。

6. パス間でASパスの長さが異なる場合は、ASパスの短い方のパスが選択されます。このステップは、bgp bestpath as-path ignore コマンドが設定されている場合は省略されます。

（注） AS パスの長さを計算する場合は、コンフェデレーション セグメントは無視され、AS セットは 1 としてカウントされます。

（注） eiBGP は、内部および外部の BGP マルチパス ピアを指定します。eiBGP では、内部および外部のパスを同時に使用できます。

7. パス間で起点が異なる場合は、起点の値が低い方のパスが選択されます。内部ゲートウェイ プロトコル（IGP）は EGP よりも低く、EGP は INCOMPLETE より低いと見なされます。

8. 該当する場合は、パスの MED が比較されます。等しくない場合は、MED の低いパスが選択されます。

   このステップが実行されるかどうかに影響するコンフィギュレーション オプションは多数あります。一般に、MED はパスが両方のパスが同じ AS にあるネイバーから受信された場合に比較され、それ以外の場合は MED 比較はスキップされます。ただし、この動作は特定のコンフィギュレーション オプションによって変更され、考慮すべきいくつかの場合があります。

   bgp bestpath med always コマンドが設定されている場合、MED 比較は、パス内のネイバー AS にかかわらず、常に実行されます。それ以外の場合、MED 比較は、次のように、比較する 2 つのパスの AS パスによって異なります。

   - パスに AS パスがない場合、または AS パスが AS_SET で始まる場合、パスは内部と見なされ、MED は他の内部パスと比較されます。
   - AS パスが AS_SEQUENCE で開始されている場合、ネイバー AS は、シーケンスの最初の AS 番号であり、MED は、同じネイバー AS を持つ他のパスと比較されます。
   - AS パスがコンフェデレーション セグメントのみを含むか、コンフェデレーション セグメントで開始されて AS_SET が続く場合、MED は、他のいずれのパスとも比較されません。ただし、**bgp bestpath med confed** コマンドが設定されている場合を除きます。その場合、パスは内部であると見なされ、MED は他の内部パスと比較されます。
   - AS パスがコンフェデレーション セグメントとそれに続く AS_SEQUENCE で開始している場合、ネイバー AS は AS_SEQUENCE の最初の AS 番号であり、MED は同じネイバー AS を持つ他のパスと比較されます。

（注） パスとともに MED 属性を受信しなかった場合、MED は 0 であると見なされます。ただし、**bgp bestpath med missing-as-worst** コマンドが設定されている場合を除きます。この場合、MED 属性が受信されていない場合、MED は最高値と見なされます。

9. パスの 1 つを外部ピアから受信し、もう 1 つを内部（またはコンフェデレーション）ピアから受信した場合は、外部ピアからのパスが選択されます。

10. パスのネクスト ホップへの IGP メトリックが異なる場合、IGP メトリックが小さい方のパスが選択されます。

11. パスの IP コスト コミュニティが等しくない場合は、IP コスト コミュニティの低いパスが最適パスとして選択されます。

12. ステップ 1 ～ステップ 10 ですべてのパス パラメータが一致している場合は、ルータ ID が比較されます。送信元属性付きでパスを受信した場合は、この属性が比較対象のルー

タ ID として使用されます。それ以外の場合は、パスの受信元ネイバーのルータ ID が使用されます。パス間でルータ ID が異なる場合は、ルータ ID の小さい方のパスが選択されます。

（注） 送信元をルータ ID として使用する場合は、2 つのパスが同じルータ ID を持つことがあります。同じピア ルータと 2 つの BGP セッションを持つこともでき、したがって、同じルータ ID を持つ 2 つのパスを受信することがあります。

13. パス間でクラスタ長が異なる場合は、クラスタ長の小さい方のパスが選択されます。クラスタ リスト属性なしでパスを受信した場合、クラスタの長さは 0 であると見なされます。
14. 最後に、IP アドレスの小さいネイバーから受信したパスが選択されます。ローカル生成されたパス（たとえば、再配布されたパス）は、ネイバー IP アドレスが 0 であると見なされます。

比較の順序

BGP 最適パス アルゴリズム実装のパート 2 では、パスの比較順序を決定します。比較順序は次のように決定されます。

1. 各グループ内のすべてのパス間で MED を比較できるように、パスがグループ分けされます。2 つのパス間で MED を比較できるかどうかは、パスのペアの比較（138 ページ）と同じルールを使用して決定されます。通常、この比較の結果は、ネイバー AS ごとに 1 グループになります。**bgp bestpath med always** コマンドが設定されている場合は、パスを含む 1 グループだけがあります。
2. 各グループ内の最適パスが決定されます。最適パスは、グループ内のすべてのパスを反復処理し、その時点までの最適なパスを追跡することによって決定されます。各パスが、この時点までの最適なパスと比較され、より適していれば新しいこの時点までの最適なパスになって、グループ内の次のパスと比較されます。
3. ステップ 2 の各グループから選択した最適パスで構成される、パスのセットを形成します。このパス セットに対してステップ 2 と同様の比較を繰り返すことによって、全体としての最適パスを選択します。

最適パスの変更の抑制

実装のパート 3 では、最適パスの変更を抑制するかどうか、つまり、新しい最適パスを使用するのか、既存の最適パスの使用を続行するのかを決定します。最適パス選択アルゴリズムが任意性を持つ部分まで、新規の最適パスと一致している場合は（ルータ ID が同一であることが前提）、引き続き既存の最適パスを使用できます。既存の最適パスの使用を続行すると、ネットワークでのチャーンを回避できます。

（注） この抑制動作は、IETF ネットワーキング ワーキング グループの draft-ietf-idr-bgp4-24.txt 資料に準拠していませんが、IETF ネットワーキング ワーキング グループの draft-ietf-idr-avoid-transition-00.txt 資料に指定されています。

この抑制動作は、**bgp bestpath compare-routerid** コマンドを設定してオフにできます。このコマンドを設定すると、新しい最適パスが常に既存の最適パスよりも優先されます。

それ以外の場合は、次の手順を使用して、最適パスの変更を抑制するかどうかが決定されます。

1. 既存の最適パスが有効でなくなった場合は、変更を抑制できません。
2. 既存または新規の最適パスを内部（またはコンフェデレーション）ピアから受信したか、ローカルで生成した（再配布によるなど）場合は、変更を抑制できません。つまり、抑制は、両方のパスを外部ピアから受信した場合のみ可能です。
3. パスを同じピアから受信した場合（通常はパスのルータ ID が同一）は、変更を抑制できません。ルータ ID は、パスのペアの比較 （138 ページ） のルールを使用して計算されます。
4. パスの重み、ローカル プリファレンス、起点、またはネクスト ホップへの IGP メトリックが異なる場合は、変更を抑制できません。このすべての値は、パスのペアの比較 （138 ページ） のルールを使用して計算されます。
5. パスの AS パス長が異なり、**bgp bestpath as-path ignore** コマンドが設定されていない場合は、変更を抑制できません。この場合もやはり、AS パスの長さは、パスのペアの比較 （138 ページ） のルールを使用して計算されます。
6. パスの MED を比較でき、MED が異なる場合は、変更を抑制できません。MED を比較できるかどうかは、パスのペアの比較 （138 ページ） で説明されている MED 値の計算とまったく同じルールによって判定されます。
7. ステップ 1 ～ステップ 6 のすべてのパス パラメータに該当しない場合は、変更を抑制できます。

BGP アップデートの生成およびアップデート グループ

BGP アップデート グループ機能により、BGP アップデートの生成がネイバー設定から分離されます。BGP アップデート グループ機能により、アウトバウンド ルーティング ポリシーに基づいて BGP アップデート グループ メンバーシップを動的に計算するアルゴリズムが導入されます。この機能に対してネットワーク オペレータによる設定は不要です。アップデート グループをベースとするメッセージ生成は自動的かつ個別に行われます。

BGP アップデート グループ

設定の変更があった場合、ルータでは、アップデート グループ メンバーシップを自動的に再計算し、変更を適用します。

BGP アップデート グループの生成を最適化するには、ネットワーク オペレータは、類似するアウトバウンド ポリシーを持つネイバーのアウトバウンド ルーティング ポリシーを同じものにしておくことを推奨します。この機能には、BGP アップデート グループを監視するためのコマンドが含まれます。

BGP コスト コミュニティの参照

コスト コミュニティ属性は、ルート ポリシーで **set extcommunity cost** コマンドを設定することにより、内部ルートに適用されます。cost community set 句は、コスト コミュニティ ID 番号（0 ～ 255）およびコスト コミュニティ番号（0 ～ 4294967295）を使用して設定されます。コスト コミュニティ番号によってパスの優先度が判断されます。最も低いコスト コミュニティ番号を持つパスが優先されます。コスト コミュニティ番号を具体的に設定していないパスには、デフォルトのコスト コミュニティ番号である 2147483647（0 ～ 4294967295 の中央値）が割り当てられ、最適パス選択プロセスにより評価されます。2 つのパスが同じコスト コミュニティ番号を使用して設定されている場合、パス選択プロセスでは最も低いコスト コミュニティ ID のパスが優先されます。このコスト拡張コミュニティ リンク属性は、拡張コミュニティ交換がイネーブルなとき、iBGP ピアに伝播します。

次のコマンドには **route-policy** キーワードが含まれています。このキーワードは、cost community set 句で設定されるルート ポリシーを適用するために使用できます。

- **aggregate-address**
- **redistribute**
- **network**

BGP ネクスト ホップの参照

RIB からのイベント通知は、クリティカルおよび非クリティカルとして分類されます。クリティカルおよび非クリティカル イベントの通知は、別々のバッチで送信されます。BGP は、次のいずれかのイベントが発生したときに通知を受けます。

- ネクスト ホップが到達不能になった。
- ネクスト ホップが到達可能になった。
- ネクスト ホップへの完全な繰り返し IGP メトリックが変更される。
- ファースト ホップの IP アドレスまたはファースト ホップのインターフェイスが変更される。
- ネクスト ホップが接続された。
- ネクスト ホップが接続解除された。

- ネクスト ホップがローカル アドレスになった。
- ネクスト ホップが非ローカル アドレスになった。

（注） 到達可能性および再帰メトリック イベントは、最適パスの再計算をトリガーします。

ただし、非クリティカル イベントが保留中であり、クリティカル イベントを読み込む要求がある場合は、非クリティカル イベントがクリティカル イベントとともに送信されます。

- クリティカル イベントは、ネクスト ホップの到達可能性（到達可能と到達不能）、接続性（接続と非接続）、および局在性（ローカルと非ローカル）に関係があります。これらのイベントの通知は遅延しません。
- 非クリティカルイベントには、IGP メトリックの変更のみが含まれます。これらのイベントは 3 秒の間隔で送信されます。メトリック変更イベントは最後の 1 つが送信されてから 3 秒後にバッチ処理され、送信されます。

BGP は、次のいずれかのイベントが発生したときに通知を受けます。

- ネクスト ホップが到達不能になった。
- ネクスト ホップが到達可能になった。
- ネクスト ホップへの完全な繰り返し IGP メトリックが変更される。
- ファースト ホップの IP アドレスまたはファースト ホップのインターフェイスが変更される。
- ネクスト ホップが接続された。
- ネクスト ホップが接続解除された。
- ネクスト ホップがローカル アドレスになった。
- ネクスト ホップが非ローカル アドレスになった。

（注） 到達可能性および再帰メトリック イベントは、最適パスの再計算をトリガーします。

クリティカルおよび非クリティカル イベントのネクスト ホップ トリガー遅延は、nexthop trigger-delay コマンドを使用して、クリティカルおよび非クリティカル イベントの最小バッチ間隔を指定するように設定できます。トリガー遅延は、アドレス ファミリに依存します。

BGP ネクストホップ トラッキング機能では、次の特性を持つルートを持つネクスト ホップだけを BGP ルートの解決に使用するように指定することができます。

- 集約ルートを回避するために、プレフィックスの長さは指定された値よりも長くなっている。

- 振動につながる可能性のあるネクスト ホップの解決にBGPルートが使用されないように、選択したリストにソース プロトコルが含まれている。

このルート ポリシーのフィルタリングが可能なのは、RIB により、ネクスト ホップを解決するルートのソース プロトコル、およびこのルートに関連付けられているマスクの長さが特定されるからです。nexthop route-policy コマンドは、ルート ポリシーを指定するために使用します。

ピアリング インターフェイスの IPv6 アドレスとしてのネクスト ホップ

BGP を使用すると、IPv4 セッションで IPv6 プレフィックスを伝送できます。IPv6 プレフィックスのネクスト ホップは、ネクスト ホップ ポリシーを使用して設定できます。ポリシーが設定されていない場合、ネクスト ホップはピアリング インターフェイスの IPv6 アドレスとして設定されます（いずれかのインターフェイスが設定されている場合は、IPv6 ネイバー インターフェイスまたは IPv6 の更新送信元インターフェイス）。

ネクスト ホップ ポリシーが設定されておらず、IPv6 ネイバー インターフェイスも IPv6 更新送信元インターフェイスも設定されていない場合は、ネクスト ホップは IPv4 射影 IPv6 アドレスになります。

範囲を指定した IPv4/VPNv4 テーブル ウォーク

処理するアドレス ファミリを判別するために、ネクスト ホップと関連付けられたゲートウェイ コンテキストを逆参照し、次に、ゲートウェイ コンテキストを調べてそのゲートウェイ コンテキストを使用しているアドレス ファミリを判別することにより、ネクスト ホップ通知が受信されます。IPv4 ユニキャストと VPNv4 ユニキャスト アドレス ファミリは、RIB 内の IPv4 ユニキャスト テーブルに登録されるため、同じゲートウェイ コンテキストを共有します。その結果、RIB から IPv4 ユニキャスト ネクスト ホップ通知を受信したときは、グローバル IPv4 ユニキャスト テーブルと VPNv4 テーブルの両方が処理されます。ネクスト ホップでマスクを保持することで、そのネクスト ホップが、IPv4 ユニキャストまたは VPNv4 ユニキャスト、あるいはその両方に属しているかどうかを示します。この範囲を指定したテーブル ウォークにより、適切なアドレス ファミリ テーブル内に処理が限定されます。

アドレス ファミリ処理の並べ替え

ソフトウェアでは、アドレス ファミリの数値に基づいてアドレス ファミリ テーブルを探索します。ネクスト ホップ通知バッチを受信すると、アドレス ファミリ処理の順序が、次の順序に並べ替えられます。

- IPv4 トンネル
- VPNv4 ユニキャスト
- VPNv6 ユニキャスト
- IPv4 ラベル付きユニキャスト
- IPv4 ユニキャスト
- IPv4 MDT
- IPv6 ユニキャスト

- IPv6 ラベル付きユニキャスト
- IPv4 トンネル
- VPNv4 ユニキャスト
- IPv4 ユニキャスト
- IPv6 ユニキャスト

ネクスト ホップ処理の新規スレッド

spkr プロセスの critical-event スレッドでは、ネクスト ホップ、双方向フォワーディング検出（BFD）、および高速外部フェールオーバー（FEF）の通知のみを処理します。この critical-event スレッドによって、BGP コンバージェンスは、大量の時間を必要とするおそれのある他のイベントによる悪影響が確実に受けなくなります。

show、clear、debug コマンド

show bgp nexthops コマンドは、ネクスト ホップ通知に関する統計情報、この通知の処理に費やした時間、および RIB に登録されている各ネクスト ホップに関する詳細を表示します。**clear bgp nexthop performance-statistics** コマンドは、モニタリングを容易にするために、ネクスト ホップの **show** コマンドの処理部分に関する累積統計情報をクリアします。**clear bgp nexthop registration** コマンドは、ネクスト ホップを RIB に非同期的に登録します。

debug bgp nexthop コマンドは、ネクスト ホップ処理の情報を表示します。**out** キーワードを指定すると、RIB に登録されている BGP のネクスト ホップに関するデバッグ情報のみが表示されます。 **in** キーワードを指定した場合は、RIB から受信したネクストホップ通知に関するデバッグ情報が表示されます。 **out** キーワードでは、RIB に送信されたネクストホップ通知に関するデバッグ情報が表示されます。

BGP ノンストップ ルーティング リファレンス

BGP NSR では、次のイベントの際のノンストップ ルーティングを実現します。

- ルート プロセッサ スイッチオーバー
- BGP または TCP でのプロセスのクラッシュまたはプロセス障害

（注） BGP NSR は、デフォルトで有効になっています。BGP NSR を無効にするには、**nsr disable** コマンドを使用します。また、無効になっている BGP NSR を有効に戻すには、**no nsr disable** コマンドを使用します。

プロセスのクラッシュまたはプロセス障害が発生した場合、NSR は **nsr process-failures switchover** コマンドが設定されている場合にのみ維持されます。アクティブなインスタンスのプロセス障害が発生した場合は、**nsr process-failures switchover** により復旧処理としてフェールオーバーが設定され、スタンバイ ルート プロセッサ（RP）またはスタンバイ分散型ルート プロセッサ（DRP）にスイッチ オーバーが行われることで、NSR が維持されます。コンフィギュレーション コマンドの一例として、RP/0/RSP0/CPU0:router(config) # nsr process-failures switchover があります。

nsr process-failures switchover コマンドは、BGP または TCP プロセスがクラッシュした場合に NSR セッションと BGP セッションの両方を維持します。この設定を行わないと、BGP プロセスまたは TCP プロセスがクラッシュした場合に BGP ネイバー セションがフラップします。この設定は、BGP ネイバーのフラップが予想される場合に BGP プロセスまたは TCP プロセスが再起動する場合は役立ちません。

l2vpn_mgr プロセスが再起動されると、NSR クライアント（te-control）は、**Ready** 状態と **Not Ready** 状態を繰り返します。これは予想される動作であり、トラフィックが損失することはありません。

ルート プロセッサ スイッチオーバーおよびインサービス システムのアップグレード（ISSU）の間、NSR は TCP と BGP の両方のステートフル スイッチオーバー（SSO）によって実現されます。

NSR では、ネットワーク内の他のルータ上でソフトウェア アップグレードを強要せず、NSR をサポートするためにピア ルータは必要ありません。

障害に起因するルート プロセッサ スイッチオーバーが発生した場合、TCP 接続および BGP セッションはトランスペアレントにスタンバイ ルートプロセッサに移行され、スタンバイ ルート プロセッサがアクティブになります。既存のプロトコル ステートは、アクティブになるスタンバイ ルート プロセッサ上で維持されて、ピアによるプロトコル ステートのリフレッシュは不要です。

ソフト再設定やポリシーの変更などのイベントにより、BGP の内部状態が変化することがあります。このようなイベントの際に、アクティブとスタンバイの BGP プロセスの間でステートの一貫性を確保するために、同期ポイントとして機能する、ポストイットの概念が導入されています。

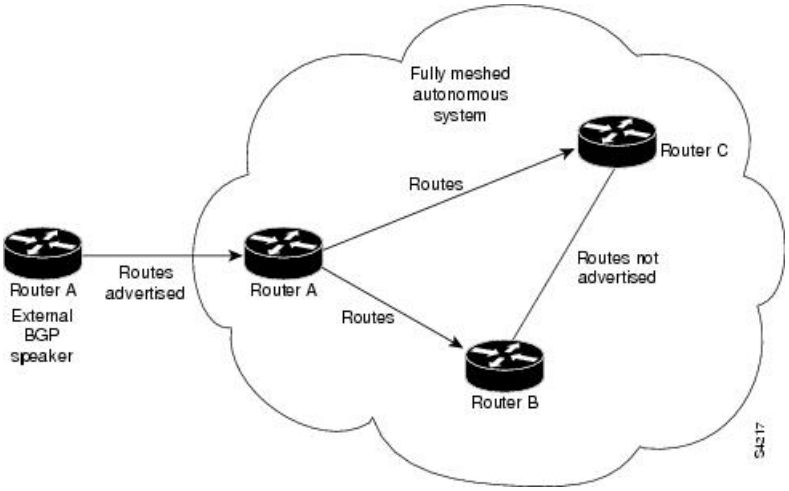
BGP NSR には次の機能があります。

- NSR 関連のアラームおよび通知
- 設定され、動作している NSR の状態は、個別に追跡される
- NSR 統計情報の収集
- **show** コマンドを使用した NSR 統計情報の表示
- XML スキーマのサポート
- アクティブとスタンバイのインスタンス間のステート同期を検証する監査メカニズム
- NSR をイネーブルおよびディセーブルにする CLI コマンド

BGP ルート リフレクタ リファレンス

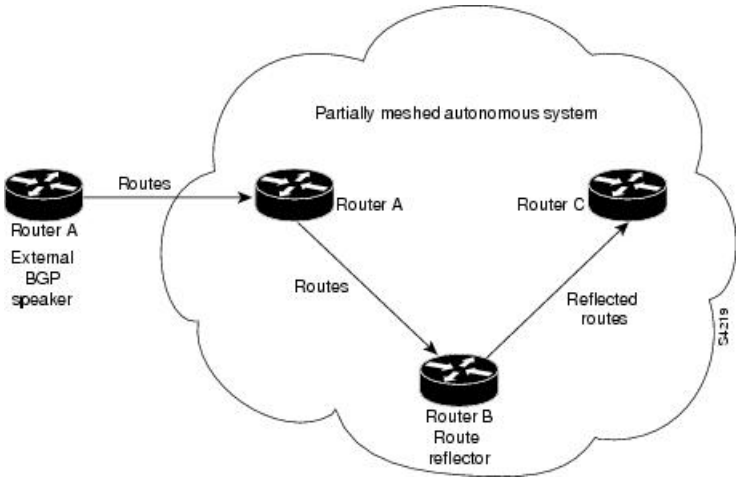
図 3 : 完全メッシュ化された 3 つの iBGP スピーカー （147 ページ） に、3 つの iBGP スピーカー（ルータ A、B、C）を持つ、単純な iBGP 設定の例を示します。ルート リフレクタがない場合、ルータ A は外部ネイバーからルートを受け取ると、そのルートをルータ B と C の両方にアドバタイズする必要があります。ルータ B と C は iBGP が学習したルートを他の iBGP スピーカーに再アドバタイズしません。これは、これらのルータが内部ネイバーから他の内部ネイバーに学習したルートを渡さないことで、ルーティング情報のループを防ぐためです。

図 *3* : 完全メッシュ化された *3* つの *iBGP* スピーカー



ルートリフレクタがある場合は、学習したルートをネイバーに渡す方法があるため、すべての iBGP スピーカーを完全にメッシュ化する必要はありません。このモデルでは、iBGP が学習したルートを一連の iBGP ネイバーに渡す役割を持つルート リフレクタとして、1 つの iBGP ピアを設定しています。図 4 : ルートリフレクタのある単純な BGP モデル （148 ページ） では、ルータ B がルートリフレクタとして設定されています。ルータ A からアドバタイズされたルートをルートリフレクタが受信すると、ルータ C にアドバタイズします。逆の場合も同じです。このスキームにより、ルータ A とルータ C 間の iBGP セッションは不要になります。

図 *4:* ルート リフレクタのある単純な *BGP* モデル



ルート リフレクタの内部ピアは、次の2種類のグループに分けられます。クライアントのピアと、自律システム内の他の全ルータ（非クライアント ピア）です。ルート リフレクタは、これらの2つのグループ間でルートを反映させます。ルート リフレクタおよびそのクライアント ピアは、クラスタを形成します。非クライアント ピアは相互に完全メッシュ構造にする必要がありますが、クライアント ピアはその必要はありません。クラスタ内のクライアントは、クラスタ外の iBGP スピーカーとは通信しません。

図 *5:* より複雑な *BGP* ルート リフレクタのモデル

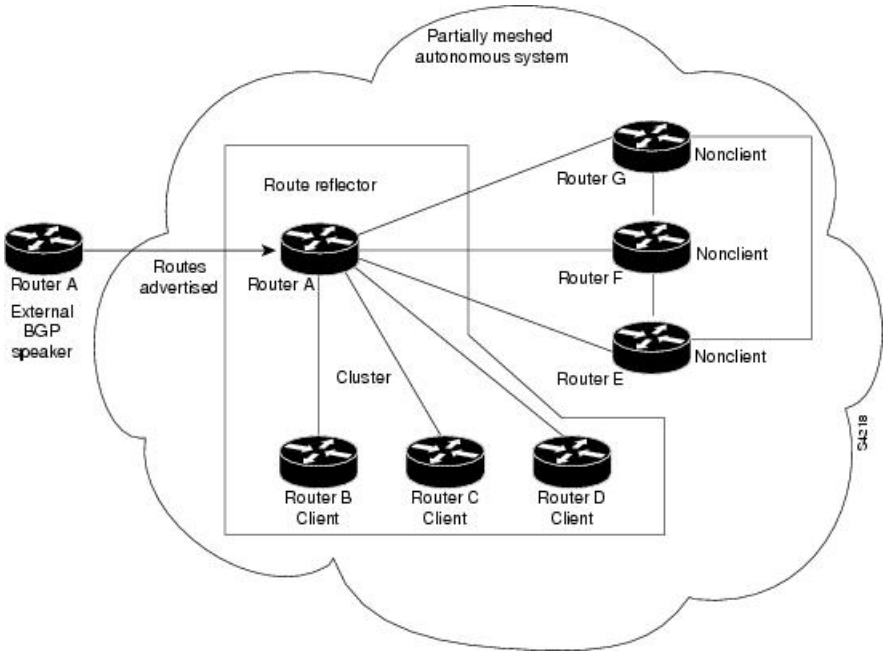


図 5 : より複雑な BGP ルート リフレクタのモデル （148 ページ） に、より複雑なルート リフレクタのスキームを示します。ルータ A は、ルータ B、C、および D があるクラスタ内のルー

トリフレクタです。ルータ E、F、および G は完全にメッシュ化された非クライアント ルータです。

ルート リフレクタがアドバタイズされたルートを受信すると、ネイバーに応じて、次のようなアクションを取ります。

- 外部 BGP スピーカーからのルートをすべてのクライアントおよび非クライアント ピアにアドバタイズします。
- 非クライアント ピアからのルートをすべてのクライアントにアドバタイズします。
- クライアントからのルートをすべてのクライアントおよび非クライアント ピアにアドバタイズします。したがって、クライアントを完全メッシュ構造にする必要はありません。

ルート リフレクタ対応の BGP スピーカーとともに、ルート リフレクタの概念に対応していない BGP スピーカーを併用することもできます。これらは、クライアントまたは非クライアント グループのメンバとなることができます。したがって、旧 BGP モデルからルート リフレクタ モデルへ、簡単に順次移行できます。たとえば、最初に、ルート リフレクタおよびいくつかのクライアントを持つ単一のクラスタを作成します。他のすべての iBGP スピーカーはルート リフレクタに対して非クライアント ピアとすることができ、クラスタを作成して徐々に追加します。

自律システムは複数のルート リフレクタを持つことができます。ルート リフレクタは、他のルート リフレクタを他の iBGP スピーカーと同様に扱います。ルート リフレクタは、他のルート リフレクタをクライアント グループまたは非クライアント グループに含むように設定できます。単純な設定では、バックボーンを多数のクラスタに分割してもかまいません。各ルート リフレクタは、非クライアント ピアとして他のルート リフレクタとともに設定されます（このため、すべてのルート リフレクタは完全メッシュ化されます）。クライアントは、所属するクラスタのルート リフレクタとだけ、iBGP セッションを維持するように設定されます。

通常、クライアントのクラスタには、ルート リフレクタが 1 つ存在します。その場合、クラスタはルート リフレクタのルータ ID で識別されます。冗長性を向上させ、シングル ポイント障害を避けるために、クラスタは複数のルート リフレクタを含むことがあります。この場合、クラスタ内のすべてのルート リフレクタにクラスタ ID を設定し、ルート リフレクタが同一クラスタ内のルート リフレクタからのアップデートを識別できるようにする必要があります。クラスタに機能を提供しているルート リフレクタはすべて完全メッシュ化され、同一のクライアントおよび非クライアント ピアのセットを持っている必要があります。

デフォルトでは、ルート リフレクタのクライアントは完全メッシュ化されている必要はなく、クライアントからのルートは他のクライアントに反映されます。ただし、クライアントが完全メッシュ化されている場合は、ルート リフレクタはルートをクライアントに反映する必要はありません。

iBGP が学習したルートが反映されるため、ルーティング情報がループする場合があります。ルート リフレクタ モデルには、ルーティングのループを防ぐ、次のようなメカニズムがあります。

- 送信元 ID は、任意で非過渡的な BGP 属性です。これは 4 バイトの属性で、ルート リフレクタにより作成されます。この属性は、ローカル自律システムのルートの送信元のルータ

ID を保持します。したがって、設定ミスによりルーティング情報が送信元に戻ってくる場合、その情報は無視されます。

- クラスタ リストは任意で非過渡的な BGP 属性です。これは、ルートが渡したクラスタ ID のシーケンスです。ルート リフレクタでは、クライアントから非クライアント ピアにルートを反映するとき（およびその逆のとき）、ローカル クラスタ ID をクラスタ リストに付加します。クラスタ リストが空の場合は、新規のクラスタ リストが作成されます。ルート リフレクタでは、この属性を使用して、設定ミスによりルーティング情報が同じクラスタにループ バックしているかどうかを識別できます。クラスタ リストにローカル クラスタ ID が見つかった場合、そのアドバタイズメントは無視されます。

iBGP マルチパス ロード シェアリングの参照

eBGP から取得した到達可能性情報を持つ複数の境界 BGP ルータがあり、ローカル ポリシーが適用されていない場合、境界ルータでは、eBGPパスを最適パスとして選択します。境界ルータでは、この最適パスをISPネットワークの内部にアドバタイズします。コアルータの場合、同じ宛先に対し複数のパスがある場合がありますが、1つのパスのみを最適パスとして選択し、そのパスを転送用に使用します。iBGP マルチパス ロード シェアリングでは、複数の等距離パス間でロード シェアリングを可能にする機能が追加されます。複数の iBGP の最適パスを設定すると、ルータでは、特定のサイトを宛先とするトラフィックを均等に負担できるようになります。iBGP のマルチパス ロード シェアリング機能は、サービス プロバイダー バックボーンを持つマルチプロトコル ラベル スイッチング（MPLS）バーチャル プライベート ネットワーク（VPN）と同様に機能します。

同じ宛先への複数のパスをマルチパスと見なすには、次の基準を満たす必要があります。

- すべての属性が同じである必要があります。属性には、重み、ローカルプリファレンス、自律システム パス（長さだけでなく属性全体）、発信元コード、Multi Exit Discriminator（MED）、および Interior Gateway Protocol（IGP）距離が含まれます。
- 各マルチパスのネクスト ホップ ルータが異なっている必要があります。

基準を満たしていて、複数のパスがマルチパスと見なされても、BGP対応ルータは、マルチパスの 1 つを最適パスに指定し、この最適パスをそのネイバーにアドバタイズします。

（注）

- マルチパス プレフィックスのネクストホップ計算の上書きは許可されていません。**next-hop-unchanged multipath** コマンドを使用すると、マルチパス プレフィックスのネクストホップ計算の上書きが無効になります。
- マルチパスの計算時に as-path オンワードを無視する機能が追加されます。 **bgp multipath as-path ignore onwards** コマンドを使用すると、マルチパスの計算時に as-path オンワードが無視されます。

L3VPN iBGP PE-CE リファレンス

プロバイダーエッジ（PE）またはカスタマーエッジ（CE）のルーティングプロトコルとして BGP を使用すると、VPN プロバイダー自律システム（AS）とカスタマー ネットワーク自律システム間の外部ピアリングとしてピエアリングセッションが設定されます。L3VPN iBGP PE-CE 機能では、PE デバイスと CE デバイスが、PE と CE 間で広く使用されている外部 BGP ピアリングの代わりに内部 ボーダー ゲートウェイ プロトコル（iBGP）としてピアリングを行って Border Gateway Protocol（BGP）ルーティング情報を交換できます。このメカニズムは、VRF ベースの CE が iBGP として設定されている各 PE デバイスで適用されます。これにより、サービス プロバイダー（SP）は、CE に自律システムのオーバーライドを設定する必要がなくなります。この機能を有効にした場合は、異なる自律システムを使用した仮想プライベート ネットワーク（VPN）サイトの設定は不要です。

neighbor internal-vpn-client コマンドを使用すると、PE デバイスが VPN クラウド全体を CE デバイスに対して内部 VPN クライアントとして動作させることができます。これらの CE デバイスは、VRF 内部の iBGP PE-CE 接続を通じて VPN クラウドに内部的に接続されます。この接続が確立されると、PE デバイスは CE-learned パスを ATTR_SET という属性内にカプセル化し、それを VPN コアからリモートの PE デバイスまで iBGP-sourced パスで伝送します。リモートの PE デバイスでは、この属性に個別の属性が割り当てられ、送信元 CE パスが抽出されてリモート CE デバイスに送信されます。

ATTR_SET はオプションの遷移属性で、受け取った CE パス属性を伝送します。ATTR_SET 属性は、次のように BGP 更新メッセージ内にエンコードされます。

```
+------------------------------+
| Attr Flags (O|T) Code = 128  |
+------------------------------+
| Attr. Length (1 or 2 octets) |
+------------------------------+
| Origin AS (4 octets)         |
+------------------------------+
| Path attributes (variable)   |
+------------------------------+
```

Origin AS は、ATTR_SET が生成される VPN カスタマーの AS です。ATTR_SET の最小長は 4 バイト、最大長は BGP 更新メッセージの必須フィールドと属性を考慮した後のパス属性でサポートされる最大値です。最大長は 3,500 バイトまでにすることをお勧めします。ATTR_SET には、属性の MP_REACH、MP_UNREACH、NEW_AS_PATH、NEW_AGGR、NEXT_HOP、および ATTR_SET 自体を含めること（ATTR_SET 内に ATTR_SET）はできません。ATTR_SET の中にこれらの属性が見つかった場合、ATTR_SET は無効と見なされ、対応するエラー処理メカニズムが呼び出されます。

MPLS VPN Carrier Supporting Carrier

Carrier Supporting Carrier（CSC）は、サービス プロバイダーの 1 つが別のサービス プロバイダーに自社のバックボーン ネットワークのセグメントの使用を許可する状況を記述した用語です。他のプロバイダーにバックボーン ネットワークのセグメントを提供するサービス プロバイダーは、バックボーン キャリアと呼ばれます。バックボーン ネットワークのセグメントを使用するサービス プロバイダーは、カスタマー キャリアと呼ばれます。

バックボーン キャリアは、ボーダー ゲートウェイ プロトコル/マルチプロトコル ラベル スイッチング（BGP/MPLS）VPN サービスを提供します。カスタマー キャリアは、次のいずれかになります。

- インターネット サービス プロバイダー（ISP）（定義上、ISP は VPN サービスを提供しません）
- BGP/MPLS VPN サービス プロバイダー

BGP をイネーブルにするように CSC ネットワークを設定して、バックボーン キャリア プロバイダー エッジ（PE）ルータとカスタマー キャリア カスタマー エッジ（CE）ルータ間のルートおよび MPLS ラベルを、複数パスを使用して転送できます。BGP を使用して IPv4 ルートと MPLS ラベル ルートを配布する利点を次に示します。

- BGPは、VPNルーティング/転送（VRF）テーブル内で内部ゲートウェイプロトコル（IGP）およびラベル配布プロトコル（LDP）の代わりをします。BGPを使用して、ルートおよびMPLS ラベルを配布できます。2 つではなく単一のプロトコルを使用すると、設定およびトラブルシューティングが簡単になります。
- BGP は、2 つの ISP を接続する場合の優先ルーティング プロトコルです。主な理由は、そのルーティング ポリシーと拡張性です。ISP では、通常、2 つのプロバイダー間で BGP を使用します。この機能を使用すると、これらの ISP は BGP を使用できます。

BGP を使用した MPLS VPN CSC 設定の詳細については、『*MPLS Configuration Guide*』のモジュールの「*Implementing MPLS Layer 3 VPNs*」を参照してください。

IPv6 プロバイダー エッジの VRF ごとおよび CE ごとのラベル

IPv6 のための VRF ごとおよび CE ごとのラベルの機能により、デフォルト VRF ごとまたは CE ネクスト ホップごとにラベルを割り当てることにより、ラベル スペースを節約できるようになります。

デフォルトでは、すべての IPv6 プロバイダー エッジ（6PE）ラベルは、プレフィックスごとに割り当てられます。VRFインスタンスに属する各プレフィックスは1つのラベルを使ってアドバタイズされます。これは、パケットのカスタマー エッジ（CE）ネクスト ホップを決定するために、VRF フォワーディング テーブルでさらにルックアップが行われる原因になります。

ただし、**per-ce** キーワードまたは **per-vrf** キーワードを指定して **label-allocation-mode** コマンドを使用すると、PE ルータ上での追加のルックアップが回避され、ラベル スペースが節約されます。

一意のカスタマー エッジ（CE）ピア ルータからアドバタイズされたすべてのルートで同じラベルを使用するように指定するには、**per-ce** キーワードを使用します。一意の VRF からアドバタイズされたすべてのルートで同じラベルを使用するように指定するには、**per-vrf** キーワードを使用します。

IPv6 ユニキャスト ルーティング

Cisco では、インターネット プロトコル バージョン 6（IPv6）の完全なユニキャスト機能を提供しています。

IPv6ユニキャストアドレスは、単一ノード上の単一インターフェイスの識別子です。ユニキャスト アドレスに送信されたパケットは、そのアドレスが示すインターフェイスに配信されます。Cisco IOS XR ソフトウェアでは、次の IPv6 ユニキャスト アドレス タイプがサポートされます。

- 集約可能グローバル アドレス
- サイトローカル アドレス
- リンクローカル アドレス
- IPv4 互換 IPv6 アドレス

IPv6 ユニキャスト アドレッシングの詳細については、『*IP Addresses and Services Configuration Guide*』を参照してください。

BGP の AS パスからのプライベート AS 番号の削除および置換

プライベート自律システム番号（ASN）は、グローバルに一意な AS 番号を保護するために、インターネット サービス プロバイダー（ISP）およびお客様のネットワークで使用されます。プライベート AS 番号は一意でないため、グローバル インターネットへのアクセスには使用できません。AS 番号はルーティング アップデートの eBGP AS パスに表示されます。プライベート ASN を使用している場合にグローバル インターネットにアクセスするには、AS パスからプライベート ASN を削除する必要があります。

パブリックな AS 番号は、InterNIC によって割り当てられ、グローバルに一意です。範囲は 1 ～ 64511 です。プライベート AS 番号は、グローバルに一意な AS 番号（有効な範囲は 64512 ～ 65535）を保護するために使用されます。プライベート AS 番号はグローバル BGP ルーティング テーブルにリークできません。プライベート AS 番号は一意ではなく、BGP 最適パスの計算には一意の AS 番号が必要であるからです。そのため、ルートが BGP ピアに伝播される前に、AS パスからプライベート AS 番号を削除する必要がある可能性があります。

外部 BGP（eBGP）では、グローバルなインターネットへのルーティングで、グローバルに一意な AS 番号を使用する必要があります。プライベート AS 番号（これは一意でない）を使用すると、グローバルなインターネットにアクセスできません。BGP の AS パスからプライベート ASN を削除および交換する機能によって、プライベート AS に属するルータがグローバルなインターネットにアクセスできるようになりました。ネットワーク管理者は、発信アップデート メッセージに含まれる AS パスからプライベート AS を削除するようにルータを設定します。場合によっては、これらの番号をローカル ルータの ASN で置き換えて、AS パス長が変化しないようにします。

AS パスからプライベート ASN を削除および交換する機能は、次のように拡張されました。

- **remove-private-as** コマンドは、AS パスにパブリックとプライベートの両方の ASN が含まれる場合も、AS パスからプライベート AS 番号を削除します。
- **remove-private-as** コマンドは、AS パスにプライベート AS 番号のみが含まれる場合も、プライベート AS 番号を削除します。このコマンドは eBGP ピアのみに適用され、その場合、eBGP ピアではローカル ルータの AS 番号が AS パスに付加されるため、長さ 0 の AS パスにはなることはありません。
- **remove-private-as** コマンドは、AS パスでコンフェデレーション セグメントの前にプライベート ASN が出現する場合でも、プライベート AS 番号を削除します。
- **replace-as** コマンドは、パスから削除されるプライベート AS 番号をローカル AS 番号に置き換えることで、AS パスを同じ長さに保ちます。

この機能は、アドレス ファミリごとにネイバーに適用できます（アドレス ファミリ コンフィギュレーション モード）。そのため、この機能をあるアドレス ファミリのネイバーには適用して、別のアドレス ファミリでは適用しないようにすることで、機能が設定されているアドレス ファミリのみのアウトバウンド側のアップデート メッセージに影響を与えることができます。

プライベート AS 番号が削除または置換されたことを確認するには、**show bgp neighbors** コマンドおよび **show bgp update-group** コマンドを使用します。

BGP アップデート メッセージのエラー処理

BGP アップデート メッセージのエラー処理によって、セッションのリセットを避けるためにエラー アップデート メッセージの処理における BGP の動作が変わります。IETF IDR *I-D:draft-ietf-idr-error-handling* で説明されているアプローチに基づいて、Cisco IOS XR BGP アップデート メッセージのエラー処理を実装すると、重大度、更新エラーが発生する可能性、属性のタイプなどの要素に基づいて、BGP 更新エラーはさまざまなカテゴリに分類されます。各カテゴリで発生したエラーは、ドラフトに沿って処理されます。セッションのリセットは、エラーの処理プロセス中は可能な限り回避されます。一部のカテゴリのエラー処理は、デフォルトの動作を有効または無効にする設定コマンドによって制御されます。

基本の BGP 仕様に応じて、不正な属性を含むアップデート メッセージを受信した BGP スピーカは、不正な属性が受信されたセッションをリセットする必要があります。セッションのリセットは、不正な属性があるルートだけでなくセッションを介して交換される他の有効なルートにも影響するので、この動作は好ましくありません。

BGP のエラー処理と属性フィルタリングの syslog メッセージ

不正な形式のアップデート パケットをルータが受信すると、ROUTING-BGP-3-MALFORM_UPDATE タイプの ios_msg がコンソールに出力されます。このレートは、すべてのネイバーで 1 分間に 1 つのメッセージになるよう制限されています。不正なパケットが「Discard Attribute」（A5）または「Local Repair」（A6）アクションの対象になった場合は、ネイバー 1 つおよびアクション 1 つごとに ios_msg メッセージが出力されます。こ

れは、ネイバーが直前の「Established」状態に到達して以降に受信した不正な形式のアップデートの数とは関係ありません。

BGP エラー処理の syslog メッセージの例を次に示します。

```
%ROUTING-BGP-3-MALFORM_UPDATE : Malformed UPDATE message received from neighbor 13.0.3.50
 - message length 90 bytes,
 error flags 0x00000840, action taken "TreatAsWithdraw".
Error details: "Error 0x00000800, Field "Attr-missing", Attribute 1 (Flags 0x00, Length
 0), Data []"
```

これは「Discard Attribute」アクションに対する BGP 属性フィルタリングの syslog メッセージの例です。

```
[4843.46]RP/0/RP0/CPU0:Aug 21 17:06:17.919 : bgp[1037]: %ROUTING-BGP-5-UPDATE_FILTERED
:
One or more attributes were filtered from UPDATE message received from neighbor 40.0.101.1
 - message length 173 bytes,
 action taken "DiscardAttr".
 Filtering details: "Attribute 16 (Flags 0xc0): Action "DiscardAttr"". NLRIs: [IPv4
Unicast] 88.2.0.0/17
```

これは「Treat-as-withdraw」アクションに対する BGP 属性フィルタリングの syslog メッセージの例です。

```
[391.01]RP/0/RP0/CPU0:Aug 20 19:41:29.243 : bgp[1037]: %ROUTING-BGP-5-UPDATE_FILTERED :
 One or more attributes were filtered from UPDATE message received from neighbor 40.0.101.1
 - message length 166 bytes,
 action taken "TreatAsWdr".
 Filtering details: "Attribute 4 (Flags 0xc0): Action "TreatAsWdr"". NLRIs: [IPv4 Unicast]
 88.2.0.0/17
```

アップデート生成のための BGP-RIB のフィードバック メカニズム

アップデート生成機能のためのボーダー ゲートウェイ プロトコル ルーティング情報ベース（BGP-RIB）のフィードバック メカニズムによって、ネットワークで不完全なルート アドバタイズメントが行われて、それによってパケット損失が発生するのを防ぐことができます。このメカニズムによって、ルートがネイバーにアドバタイズされる前にローカルに組み込まれるようになります。

BGP は RIB からのフィードバックを待ちます。このフィードバックには、BGP によって RIB に組み込まれたルートが、BGP がネイバーにアップデートを送信する前に転送情報ベース（FIB）に組み込まれたことが示されています。RIB は BCDL のフィードバック メカニズムを使用して、そのバージョンのルートが FIB によって使用されたかを判断し、BGP をそのバージョンで更新します。BGP がアップデートを送信するのは、FIB が組み込んだバージョン以下のバージョンのルートだけです。この選択的な更新によって、BGP が不完全なアップデートを送信しないようになり、ルータのリロード、LC OIR、または代替パスが使用可能になるリンクフラップ後にデータ プレーンがプログラミングされる前であっても、トラフィックの引き込みが行われるようになります。

BGP が RIB に組み込んだルートが FIB に組み込まれたことを示す RIB からのフィードバックを BGP が待機し、その後で BGP がネイバーにアップデートを送信するように設定するには、ルータ アドレスファミリ IPv4 またはルータ アドレスファミリ VPNv4 コンフィギュレーション モードで **update wait-install** コマンドを使用します。**show bgp**、**show bgp neighbors**、および **show bgp process performance-statistics** コマンドを実行すると、update wait-install 設定の情報が表示されます。

ユーザ定義の Martian チェック

このソリューションによって、次の IP アドレス プレフィックスに対する Martian チェックを無効化できます。

- IPv4 アドレス プレフィックス
  - 0.0.0.0/8
  - 127.0.0.0/8
  - 224.0.0.0/4
- IPv6 アドレス プレフィックス
  - ::
  - ::0002 - ::ffff
  - ::ffff:a.b.c.d
  - fe80:xxxx
  - ffxx:xxxx