

Chapter 28

BGP Configuration Guidelines

To configure BGP, you can include statements at the [edit protocols bgp] hierarchy level in the configuration. Three portions of the bgp statement—the portions in which you configure global BGP, group-specific, and peer-specific options—contain many of the same statements. The following simplified version of the bgp statement omits these repeated statements so as to present a high-level, readable overview.

```
protocols {
  bgp {
    numerous global BGP statements
    group group-name {
      peer-as autonomous-system;
      type type;
      allow [network/mask-length];
      numerous group-specific statements
      neighbor address {
        numerous peer-specific statements
      }
    }
  }
}
```

For a list of global BGP statements, see “Define BGP Global Properties” on page 305. For a list of group-specific statements, see “Define Group Properties” on page 308. For a list of peer-specific statements, see “Define Peer Properties” on page 309.

Many of the global BGP, group-specific, and peer-specific statements are identical. For statements that you can configure at more than one level in the hierarchy, the more-specific statement overrides the less-specific statement. That is, a group-specific statement overrides a global BGP statement, and a peer-specific statement overrides a global BGP or group-specific statement.

By default, BGP is disabled.

This chapter describes the following tasks for configuring BGP:

Minimum BGP Configuration on page 303

Enable BGP on page 304

Modify the Hold-Time Value on page 314

Configure Authentication on page 315

Open a Peer Connection Passively on page 316

- Configure the Local IP Address on page 316
- Configure the Multiple Exit Discriminator Metric on page 316
- Control the Aggregator Path Attribute on page 319
- Choose the Protocol Used to Determine the Next Hop on page 320
- Configure an EBGP Multihop Session on page 320
- Configure the BGP Local Preference on page 320
- Control Route Preference on page 321
- Configure Routing Table Path Selection on page 322
- Configure BGP to Select Multiple EBGP Paths on page 323
- Configure a Local AS on page 324
- Remove Private AS Numbers from AS Paths on page 327
- Configure Route Reflection on page 328
- Enable Route Flap Damping on page 333
- Enable Multiprotocol BGP on page 334
- Configure BGP Routing Policy on page 336
- Configure BGP to Log Syslog Messages on page 340
- Describe BGP Router Configuration on page 340
- Trace BGP Protocol Traffic on page 341

Minimum BGP Configuration

For BGP to run on the router, you must define the local AS number, configure at least one group, and include information about at least one peer in the group (the peer's IP address and AS number). There are several possible ways you can configure this information, a few of which are shown in this section.

Configure a BGP group, specify the group type, and configure an explicit peer:

```
[edit]
routing-options {
  autonomous-system autonomous-system;
}
protocols {
  bgp {
    group group-name {
      peer-as autonomous-system;
      type type;
      neighbor address;
    }
  }
}
```

Configure a BGP group and type and allow all BGP systems to be peers:

```
[edit]
routing-options {
  autonomous-system autonomous-system;
}
protocols {
  bgp {
    group group-name {
      type type;
      peer-as autonomous-system;
      allow all;
    }
  }
}
```



Note

When you configure BGP on an interface, you must also include the family inet statement at the [edit interfaces *interface-name* unit *logical-unit-number*] hierarchy level. For more information about the family inet statement, see the *JUNOS Internet Software Configuration Guide: Interfaces and Chassis*.

Enable BGP

To enable BGP on the router, you do the following:

- Specify the Local Router's AS Number on page 304 (required)
- Define an AS Confederation and Its Members on page 304 (optional)
- Assign a BGP Identifier on page 305 (optional)
- Define BGP Global Properties on page 305 (optional)
- Define BGP Groups and Peers on page 306 (required)

For examples of enabling BGP, see “Examples: Enable BGP” on page 310

Specify the Local Router's AS Number

Each router running BGP must be configured with its AS number. This number is included in the local AS number field in BGP open messages, which are sent between BGP peers to establish a connection.

To specify an AS number, include the `autonomous-system` statement at the [edit routing-options] hierarchy level:

```
[edit routing-options]
autonomous-system autonomous-system <loops number>;
```

For more information about configuring the AS number, see “Configure the AS Number” on page 121.

Define an AS Confederation and Its Members

To enable the local system to participate as a member of an AS confederation, you must define the AS confederation identifier and specify the AS numbers that are members of the confederation. To do this, include the confederation statement at the [edit routing-options] hierarchy level:

```
[edit routing-options]
confederation confederation-autonomous-system members [autonomous-system];
```

For more information about configuring confederations, see “Configure AS Confederation Members” on page 122.

For a confederation configuration example, see “Examples: Enable BGP” on page 310.

Assign a BGP Identifier

Each router running BGP must have a BGP identifier. This identifier is included in the BGP identifier field of open messages, which are sent between two BGP peers when establishing a BGP session.

Explicitly assigning a BGP identifier is optional. If you do not assign one, the IP address of the first interface encountered in the router is used.

To assign a BGP identifier explicitly, include the `router-id` statement at the [edit routing-options] hierarchy level:

```
[edit routing-options]
router-id address;
```

For more information, see “Configure the Router Identifier” on page 122.

Define BGP Global Properties

To define BGP global properties, which apply to all BGP groups and peers, include one or more of the following statements at the [edit protocols bgp] hierarchy level. These statements are explained in separate sections.

```
[edit protocols bgp]
advertise-inactive;
authentication-key key;
cluster cluster-identifier;
damping;
description text-description;
family inet {
  (any | multicast | unicast) {
    prefix-limit {
      maximum number;
      teardown <percentage>;
    }
  }
}
export [policy-names];
hold-time seconds;
import [policy-names];
keep (all | none);
local-address address;
local-as autonomous-system <private>;
local-preference local-preference;
log-updown;
metric-out metric;
multihop <tth-value>;
no-aggregator-id;
no-client-reflect;
out-delay seconds;
passive;
path-selection (cisco-non-deterministic);
peer-as autonomous-system;
preference preference;
remove-private;
```

```

traceoptions {
  file name <replace> <size size> <files number> <no-stamp>
    <(world-readable | no-world-readable)>;
  flag flag <flag-modifier> <disable>;
}

```

Define BGP Groups and Peers

A BGP system must know which routers are its peers (neighbors). You define the peer relationships explicitly by configuring the neighboring routers that are the peers of the local BGP system. After peer relationships have been established, the BGP peers exchange update messages to advertise network reachability information.

You arrange BGP routers into groups of peers. Different peer groups must have different group types, AS numbers, or router reflector cluster identifiers.

Each group must contain at least one peer.

To define BGP groups and peers, you can do the following:

- Define a Group with Static Peers on page 306

- Define a Group with Dynamic Peers on page 307

- Define the Group Type on page 307

- Specify the Peer's AS Number on page 307

- Define Group Properties on page 308

- Define Peer Properties on page 309

Define a Group with Static Peers

To define a BGP group that recognizes only the specified BGP systems as peers, statically configure all the system's peers by including one or more `neighbor` statements at the `[edit protocols bgp group group-name]` hierarchy level. The peers on at least one side of each BGP connection must be configured statically.

```

[edit protocols bgp]
group group-name {
  peer-as autonomous-system;
  type type;
  neighbor address;           # One "neighbor" statement for each peer
}

```

Define a Group with Dynamic Peers

To define a BGP group in which the local system's peers are dynamic and change over time, include the allow statement at the [edit protocols bgp group *group-name*] hierarchy level. To recognize all BGP systems as peers, include the allow all statement. To recognize BGP systems within specified address ranges, specify a set of addresses in the allow *network/mask-length* statement.

```
[edit protocols bgp]
group group-name {
  peer-as autonomous-system;
  type type;
  (allow [network /mask-length...] | allow all);
}
```

Define the Group Type

When configuring a BGP group, you can indicate whether the group is an internal BGP (IBGP) group or an external BGP (EBGP) group. All peers in an IBGP group are in the same AS, while peers in an EBGP group are in different ASs and normally share a subnet.

To configure an IBGP group, which allows intra-AS BGP routing, include the following form of the type statement at the [edit protocols bgp group *group-name*] hierarchy level:

```
[edit protocols bgp group group-name]
type internal;
```

To configure an EBGP group, which allows inter-AS BGP routing, include the following form of the type statement at the [edit protocols bgp group *group-name*] hierarchy level:

```
[edit protocols bgp group group-name]
type external;
```

Specify the Peer's AS Number

When configuring a peer, you must specify the peer system's AS. To do this, include the peer-as statement:

```
peer-as autonomous-system;
```

You can configure the peer's AS globally for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

For EBGP, the peer is in another AS, so the AS number you specify in the peer-as statement must be different from the local router's AS number, which you specify in the autonomous-system statement (at the [edit routing-options] hierarchy level). For IBGP, the peer is in the same AS, so the two AS numbers that you specify in the autonomous-system and peer-as statements must be the same.

Define Group Properties

To define group-specific properties, include one or more of the following statements at the [edit protocols bgp group *group-name*] hierarchy level. The statements are explained in separate sections.

```
[edit protocols bgp group group-name]
advertise-inactive;
allow [network/mask-length];
authentication-key key;
cluster cluster-identifier;
damping;
description text-description;
export [policy-names];
family inet {
  (any | multicast | unicast) {
    prefix-limit {
      maximum number;
      teardown <percentage>;
    }
  }
}
hold-time seconds;
import [policy-names];
keep (all | none);
local-address address;
local-as autonomous-system <private>;
local-preference local-preference;
log-updown;
metric-out metric;
multihop <tth-value>;
multipath;
neighbor address {
  peer-specific options;
}
no-aggregator-id;
no-client-reflect;
out-delay seconds;
passive;
peer-as autonomous-system;
preference preference;
protocol protocol;
remove-private;
traceoptions {
  file name <replace> <size size> <files number> <no-stamp>
  <(world-readable | no-world-readable)>;
  flag flag <flag-modifier> <disable>;
}
type type;
```

Define Peer Properties

When you use the `neighbor` statement to configure BGP peers statically, you also can define peer-specific properties at the `[edit protocols bgp group group-name neighbor address]` hierarchy level. These options are explained in separate sections.

```
[edit protocols bgp group group-name neighbor address]
advertise-inactive;
authentication-key key;
cluster cluster-identifier;
damping;
description text-description;
export [policy-names];
family inet {
  (any | multicast | unicast) {
    prefix-limit {
      maximum number;
      teardown <percentage>;
    }
  }
}
hold-time seconds;
import [policy-names];
keep (all | none);
local-address address;
local-as autonomous-system <private>;
local-preference preference;
log-updown;
metric-out metric;
multihop <ttl-value>;
multipath;
no-aggregator-id;
no-client-reflect;
out-delay seconds;
passive;
peer-as autonomous-system;
preference preference;
traceoptions {
  file name <replace> <size size> <files number> <no-stamp>
    <(world-readable | no-world-readable)>;
  flag flag <flag-modifier> <disable>;
}
```

Examples: Enable BGP

Enable BGP and define an EBGP group that recognizes all BGP systems in AS 56 as peers:

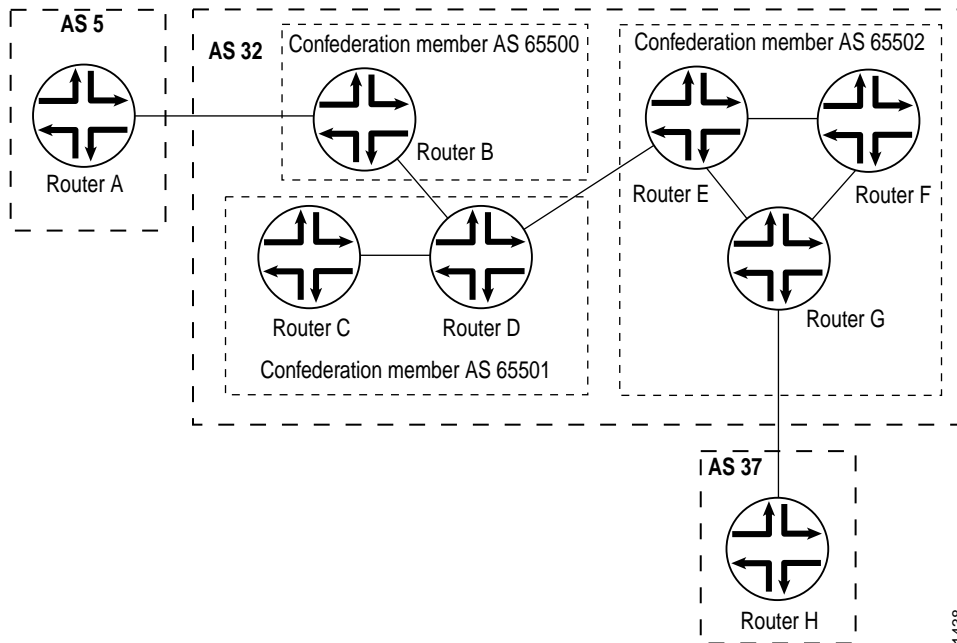
```
[edit]
routing-options {
  autonomous-system 23;
}
protocols {
  bgp {
    group 23 {
      type external;
      peer-as 56;
      allow 0.0.0.0/0;
    }
  }
}
```

Enable BGP and define an IBGP group that recognizes only the specified addresses as BGP peers:

```
[edit]
routing-options {
  autonomous-system 23;
  router-id 10.0.0.1;
}
protocols {
  bgp {
    group 23 {
      type internal;
      peer-as 23;
      neighbor 10.0.0.2;
      neighbor 10.0.0.3;
    }
  }
}
```

Configure a BGP confederation. Figure 9 illustrates the confederation topology used in this example. For AS 32 to be a valid confederation, all routers in the AS must be members of the confederation. So, for example, Router B must have a confederation member AS number as well as a confederation AS number. Within a confederation, the links between the confederation member ASs must be EBGP links, not IBGP links.

Figure 9: Example BGP Confederation Topology



1438

On Router A:

```
[edit]
routing-options {
  autonomous-system 5;
}
protocols {
  bgp {
    group AtoB {
      type external;
      peer-as 32;
      neighbor 10.0.0.2;
    }
  }
}
```


On Router E:

```
[edit]
routing-options {
  autonomous-system 65502;
  confederation 32 members [65500 65501 65502];
}
protocols {
  bgp {
    group EtoD {
      type external;
      peer-as 65501;
      neighbor 10.0.10.4;
    }
    group EtoFandG {
      type internal;
      neighbor 10.0.30.2;
      neighbor 10.0.30.5;
    }
  }
}
```

On Router F:

```
[edit]
routing-options {
  autonomous-system 65502;
  confederation 32 members [65500 65501 65502];
}
protocols {
  bgp {
    group FtoEandG {
      type internal;
      neighbor 10.0.30.3;
      neighbor 10.0.30.7;
    }
  }
}
```



On Router G:

```
[edit]
routing-options {
  autonomous-system 65502;
  confederation 32 members [65500 65501 65502];
}
protocols {
  bgp {
    group GtoH {
      type external;
      peer-as 37;
      neighbor 10.0.40.1;
    }
    group GtoEandF {
      type internal;
      neighbor 10.0.30.4;
      neighbor 10.0.30.5;
    }
  }
}
```

On Router H:

```
[edit]
routing-options {
  autonomous-system 37;
}
protocols {
  bgp {
    group HtoG {
      type external;
      peer-as 32;
      neighbor 10.0.30.8;
    }
  }
}
```

Modify the Hold-Time Value

The hold time is the maximum number of seconds allowed to elapse between the time that a BGP system receives successive keepalive or update messages from a peer. When establishing a BGP connection with the local router, a peer sends an open message, which contains a hold-time value. BGP on the local router uses the smaller of either the local hold-time value or the peer's hold-time value received in the open message as the hold time for the BGP connection between the two peers.

The default hold-time value is 90 seconds. A value of 0 means that no keepalive messages are sent. You might want to set the hold-time value to 0 when using BGP over media that provide a reliable indication of loss of link-layer connectivity; for example, switched virtual circuit (SVC) subnetworks.

To modify the hold-time value on the local BGP system, include the hold-time statement:

```
hold-time seconds;
```

You can configure the hold-time value globally for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

The hold time is three times the interval at which keepalive messages are sent.

Configure Authentication

All BGP protocol exchanges can be authenticated to guarantee that only trusted routers participate in the AS's routing. By default, authentication is disabled on the router. You can configure MD5 authentication on the router. The MD5 algorithm creates an encoded checksum that is included in the transmitted packet. The receiving router uses an authentication key (password) to verify the packet's MD5 checksum.

To configure authentication, include the authentication-key statement:

```
authentication-key key;
```

You can configure authentication globally for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level) or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level). For routing instances, include the statement at the [edit routing-instances *routing-instance-name* protocols bgp] hierarchy level, [edit routing-instances *routing-instance-name* protocols bgp group *group-name*] hierarchy level, and the [edit routing-instances *routing-instance-name* protocols bgp group *group-name* neighbor *address*] hierarchy level. If you configure authentication for all peers, each individual peer in that group inherits the group's authentication.

The key (password) can be up to 255 characters long. Characters can include any ASCII strings. If you include spaces, enclose all characters in quotation marks (double quotes).

When configuring authentication for all peers in a group, you cannot include the allow statement in the configuration because BGP keys require a destination address.

Open a Peer Connection Passively

You can configure a router not to send Open requests to a peer. Once you configure the router to be passive, the router does not originate the TCP connection. However, when the router receives a connection from the peer and an Open message, it replies with another BGP Open message. Each router declares its own capabilities.

To configure the router so that it does not send open requests to a peer, include the passive statement:

```
passive;
```

You can configure passive connections globally for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level). For routing instances, include the statement at the [edit routing-instances *routing-instance-name* protocols bgp] hierarchy level, [edit routing-instances *routing-instance-name* protocols bgp group *group-name*] hierarchy level, and the [edit routing-instances *routing-instance-name* protocols bgp group *group-name* neighbor *address*] hierarchy level.

Configure the Local IP Address

You can explicitly specify the address of the local end of a BGP session. You generally do this to explicitly configure the system's IP address from BGP's point of view. Typically, an IP address is assigned to a loopback interface, and that IP address is configured here. This address is used to accept incoming connections to the peer and to establish connections to the remote peer. To assign a local address, include the local-address statement:

```
local-address address;
```

You can configure the local BGP address globally for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

Configure the Multiple Exit Discriminator Metric

The BGP multiple exit discriminator (MED, or MULTI_EXIT_DISC) is an optional path attribute that can be included in BGP update messages. This attribute is used on external BGP links (that is, on inter-AS links) to select among multiple exit points to a neighboring AS. The MED attribute has a value that is referred to as a *metric*. If all other factors in determining an exit point are equal, the exit point with the lowest metric is preferred.

If a MED is received over an external BGP link, it is propagated over internal links to other BGP systems within the AS.

BGP update messages include a MED metric if the route was learned from BGP and already had a MED metric associated with it, or if you configure the MED metric in the configuration file in one of the following ways:

Define a MED Metric Directly on page 317

Use Routing Policy to Define a MED Metric on page 318

For configuration examples, see “Examples: Configure the MED Metric” on page 318.

A MED metric is advertised with a route according to the following general rules:

A more specific metric overrides a less specific metric. That is, a group-specific metric overrides a global BGP metric and a peer-specific metric overrides a global BGP or group-specific metric.

A metric defined with routing policy overrides a metric defined with the metric-out statement.

If any metric is defined, it overrides a metric received in a route.

If the received route did not have an associated MED metric, and if you did not explicitly configure a metric, no metric is advertised.

For a description of the algorithm used to determine the active path, see “How the Active Route Is Determined” on page 6.

Define a MED Metric Directly

To directly configure a MED metric to advertise in BGP update messages, include the metric-out statement:

```
metric-out (metric | minimum-igp | igp);
```

You can configure a MED metric globally for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

metric is the primary metric on all routes sent to peers. It can be a value in the range 0 through $2^{32} - 1$.

Specify *minimum-igp* to set the metric to the minimum metric value calculated in the IGP. If a newly calculated metric is greater than the minimum metric value, the metric value remains unchanged. If a newly calculated metric is lower, the metric value is lowered to that value.

Specify *igp* to set the metric to the most recent metric value calculated in the IGP.

Use Routing Policy to Define a MED Metric

To use routing policy to define a MED metric to advertise, define the routing policy with the `policy-statement` statement at the `[edit policy-options]` hierarchy level, and then apply the filter using the `import` and `export` statements when configuring BGP.

When defining the routing policy filter, include an action that specifies the desired metric value:

```
[edit policy-options]
policy-statement policy-name {
  term term-name {
    from {
      match-conditions;
      route-filter destination-prefix match-type <actions>;
      prefix-list name;
    }
    to {
      match-conditions;
    }
    then actions;
  }
}
```

For information about defining routing policy, see “Configure Routing Policy” on page 35. For information about applying filters in BGP, see “Configure BGP Routing Policy” on page 336.

Examples: Configure the MED Metric

Set the MED metric to 20 for all routes advertised in BGP update messages except for those sent to the peer system 192.168.0.1; the MED for this peer is 10:

```
[edit]
routing-options {
  router-id 10.0.0.1;
  autonomous-system 23;
}
protocols {
  bgp {
    metric-out 20;
    group 23 {
      type external;
      peer-as 56;
      neighbor 192.168.0.1 {
        traceoptions {
          file bgp-log-peer;
          flag packets;
        }
        log-updown;
        metric-out 10;
      }
    }
  }
}
```

Set the MED metric to 20 for all routes from a particular community:

```
[edit]
routing-options {
  router-id 10.0.0.1;
  autonomous-system 23;
}
policy-options {
  policy-statement from-otago {
    from community otago;
    then metric 20;
  }
  community otago members [56:2379 23:46944];
}
protocols {
  bgp {
    import from-otago;
    group 23 {
      type external;
      peer-as 56;
      neighbor 192.168.0.1 {
        traceoptions {
          file bgp-log-peer;
          flag packets;
        }
        log-updown;
      }
    }
  }
}
```

Control the Aggregator Path Attribute

The JUNOS implementation of BGP performs route aggregation, which is the process of combining the characteristics of different routes so that only a single route is advertised. Aggregation reduces the amount of information that BGP must store and exchange with other BGP systems.

BGP adds the aggregator path attribute to BGP update messages. This attribute contains the local system's AS number and IP address (router ID).

To prevent different routers within an AS from creating aggregate routes that contain different AS paths, set the IP address in the aggregator path attribute to 0 by including the `no-aggregator-id` statement.

```
no-aggregator-id;
```

You can control the aggregator path attribute globally for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

Choose the Protocol Used to Determine the Next Hop

By default, BGP uses the active routes determined from all IGP's when resolving routes to next hops. To limit the IGP's that BGP uses, include the protocol statement, specifying the protocol as isis or ospf:

```
protocol protocol;
```

You can control the protocol globally for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

Configure an EBGP Multihop Session

If an EBGP peer is more than one hop away from the local router, you must specify the next hop to the peer so that the two systems can establish a BGP session. This type of session is called a *multihop* BGP session. To configure a multihop session, include the multihop statement:

```
multihop <ttl-value>;
```

You can configure EBGP multihop sessions for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

To configure the maximum TTL value for the TTL in the IP header of BGP packets, specify *ttl-value*. If you do not specify a TTL value, the system's default maximum TTL value is used.

Configure the BGP Local Preference

Internal BGP sessions use a metric called the *local preference*, which is carried in internal BGP update packets in the path attribute LOCAL_PREF. This metric indicates the degree of preference for an external route. The route with the highest local preference value is preferred.

The LOCAL_PREF path attribute is always advertised to internal BGP peers and to neighboring confederations. It is never advertised to external BGP peers. The default behavior is to not modify the LOCAL_PREF path attribute if it is present.



The LOCAL_PREF path attribute applies at export time only.

Note

By default, if a received route contains a LOCAL_PREF path attribute value, the value is not modified. If a BGP route is received without a LOCAL_PREF attribute, the route is handled locally (that is, it is stored in the routing table and advertised by BGP) as if it were received with a LOCAL_PREF value of 100. A non-BGP route that is advertised by BGP is advertised with a LOCAL_PREF value of 100 by default.

To change the local preference metric advertised in the path attribute, include the local-preference statement, specifying a value from 0 through 4294967295 ($2^{32} - 1$):

```
local-preference local-preference;
```

You can configure the local preference metric for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

Control Route Preference

When the JUNOS software determines a route's preference to become the active route, it selects the route with the lowest preference as the active route and installs this route into the forwarding table. By default, the routing software assigns a preference of 170 to routes that originated from BGP. Of all the routing protocols, BGP has the highest default preference value, which means that routes learned by BGP are the least likely to become the active route. (For more information about preferences, see "Route Preferences" on page 5.)

To modify the default BGP preference value, include the preference statement, specifying a value from 0 through 4294967295 ($2^{32} - 1$):

```
preference preference;
```

You can configure the preference value for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

Examples: Control Route Preference

Assign a preference of 160 to routes learned from the BGP system 192.168.1.1. The routing protocol process will prefer these routes over routes learned from other BGP systems, which have the default preference of 170.

```
[edit]
routing-options {
  autonomous-system 23;
}
protocols {
  bgp {
    group 23 {
      type external;
      peer-as 56;
      neighbor 192.168.1.1 {
        preference 160;
      }
    }
  }
}
```

Assign a preference of 140 to all routes learned by BGP systems. Because the default OSPF preference is 150, BGP routes will be preferred over those learned from OSPF.

```
[edit]
routing-options {
  autonomous-system 23;
}
protocols {
  bgp {
    preference 140;
    group 23 {
      type external;
      peer-as 56;
      neighbor 192.168.1.1;
    }
  }
}
```

Configure Routing Table Path Selection

By default, only the MEDs of routes that have the same peer ASs are compared. You can configure routing table path selection options to get different behaviors. To configure routing table path selection behavior, include the path-selection statement at the [edit protocols bgp] hierarchy level:

```
[edit protocols bgp]
path-selection (cisco-non-deterministic | always-compare-med);
```

Routing table path selection can be configured in one of two ways:

Using the same nondeterministic behavior as does the Cisco IOS software (cisco-non-deterministic). This behavior has two effects:

The active path is always first. All nonactive, but eligible, paths follow the active path and are maintained in the order in which they were received, with the most recent path first. Ineligible paths remain at the end of the list.

When a new path is added to the routing table, path comparisons are made without removing from consideration those paths that should never be selected because those paths lose the MED tie-breaking rule.

These two effects cause the system to only sometimes compare the MEDs between paths that it should otherwise compare. Because of this, we recommend that you do not configure nondeterministic behavior.

To always compare MEDs whether or not the peer ASs of the compared routes are the same (always-compare-med).

For an example of always compare MEDs, see “Example: Always Compare MEDs” on page 323.

For a description of the algorithm used to determine the active path, see “How the Active Route Is Determined” on page 6.

Example: Always Compare MEDs

In this example, paths learned from 208.197.169.15 have their MED values compared against 4 plus the MED values of the same paths learned from 208.197.169.14.

```
[edit]
protocols {
  bgp {
    path-selection always-compare-med;
    group ref {
      type external;
      import math;
      peer-as 10458;
      neighbor 208.197.169.14;
    }
    group ref {
      type external;
      peer-as 10;
      neighbor 208.197.169.15;
    }
  }
}

policy-options {
  policy-statement math {
    then {
      metric add 4;
    }
  }
}
```

Configure BGP to Select Multiple EBGPath Paths

You can configure BGP to select multiple, nonmultihop EBGPath paths as active paths. Selecting multiple paths allows EBGPath peerings to load-balance traffic across an AS–confederation boundary. The JUNOS BGP multipath supports the following:

- Load-balancing across multiple links between two routers belonging to different ASs

- Load-balancing across a common subnet or multiple subnets to different routers belonging to the same peer AS

- Load-balancing across multiple links between two routers belonging to different external confederation peers

- Load-balancing across a common subnet or multiple subnets to different routers belonging to external confederation peers

To configure a BGP multipath, include the multipath statement:

```
multipath;
```

You can configure BGP multipath for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

Configure a Local AS

You can configure BGP with a different local AS number for each EBGP session, which allows BGP to configure a local AS for each EBGP session. Configuring a local AS simulates a virtual AS for the router. The AS paths for the routes from that EBGP peer have the configured local-as prepended before the peer AS for that session. This is useful if ISP A has acquired another ISP B, but does not want to change the configurations of ISP B's customers' routers. ISP B's AS is the AS that is configured as the local AS.

To configure a local AS, include the local-as statement:

```
local-as autonomous-system <private>;
```

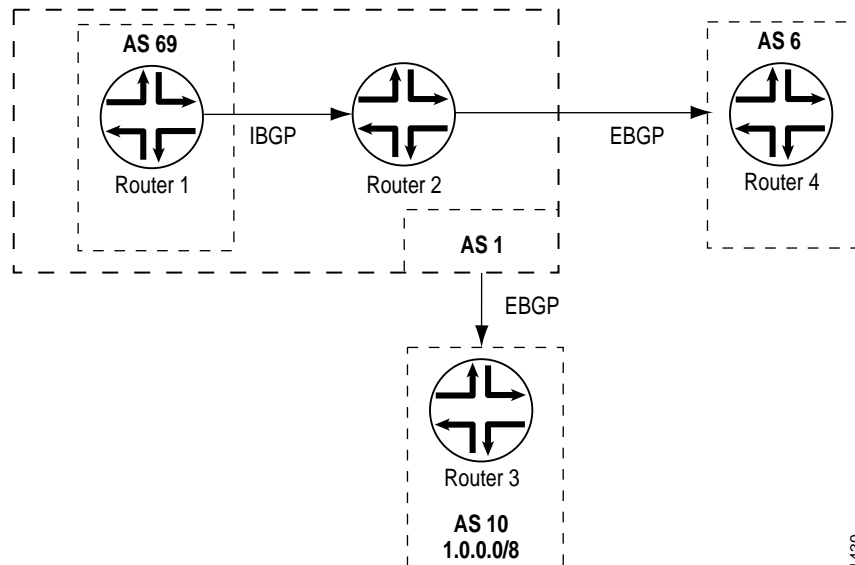
You can set the local AS path for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

If you include the private keyword, the local AS is not prepended before the peer AS. This means that the AS paths do not show details of such a configuration, and ISP A's EBGP peers and IBGP peers do not see any difference from before the local AS configuration.

Examples: Configure Local AS

Configure Router 1 to use AS 1 as its local address for an EBGP peering with Router 3 (AS 10), as illustrated in Figure 10:

Figure 10: Local AS Configuration



1439

On Router 1:

```
routing-options {  
  autonomous-system 65534;  
}  
protocols {  
  bgp {  
    group internal-AS65534 {  
      type internal;  
      local-address 10.255.245.67;  
      neighbor 10.1.2.3;  
    }  
  }  
}
```

On Router 2:

```
routing-options {  
  autonomous-system 65534;  
}  
protocols {  
  bgp {  
    group internal-AS65534 {  
      type internal;  
      local-address 10.1.2.3;  
      neighbor 10.255.245.67;  
    }  
    group external-AS10 {  
      type external;  
      peer-as 10;  
      neighbor 192.168.129.50 {  
        local-as 1;  
      }  
    }  
    group external-AS6 {  
      type external;  
      peer-as 6;  
      neighbor 192.168.129.193;  
    }  
  }  
}
```

On Router 3:

```

routing-options {
  autonomous-system 10;
  static {
    route 1.0.0.0/8 nexthop 10.10.10.1;
  }
}
protocols {
  bgp {
    group external-AS1 {
      type external;
      export redist-static;
      peer-as 1;
      neighbor 192.168.129.49;
    }
  }
}
policy-options {
  policy-statement redist-static {
    from {
      protocol static;
      route-filter 1.0.0.0/8 exact;
    }
    then accept;
  }
}

```

On Router 4:

```

routing-options {
  autonomous-system 6;
}
protocols {
  bgp {
    group external-AS65534 {
      peer-as 65534;
      neighbor 192.168.129.194;
    }
  }
}

```

In this configuration, Router 3 announces route 1.0.0.0/8 to Router 2. The following is the show route output for Router 2, Router 1, and Router 4:

user@Router 2> show route 1/8 detail

```

inet.0: 97 destinations, 97 routes (96 active, 0 holddown, 4 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

1.0.0.0/8 (2 entries, 1 announced)

```

```

 *BGP Preference: 170/-101
   Nexthop: 192.168.129.50 via so-2/0/0.0, selected
   State: <Active Ext>
   Local AS: 65534 Peer AS: 10
   Age: 4:25
   Task: BGP_10_1.192.168.129.50+179
   Announcement bits (3): 1-BGP0.0.0.0+179 2-KRT 5-BGP_Sync_Any
   AS path: 10 I
   Localpref: 100
   Router ID: 10.1.2.4

```

user@Router 1> show route 1/8 detail

inet.0: 114 destinations, 114 routes (113 active, 0 holddown, 1 hidden)

+ = Active Route, - = Last Active, * = Both

1.0.0.0/8 (1 entry, 1 announced)

```
*BGP Preference: 170/-101
Source: 10.1.2.3
Nexthop: 192.168.129.33 via so-6/0/0.0, selected
Nexthop: 192.168.129.37 via so-6/1/0.0
State: <Active Int Ext>
Local AS: 65534 Peer AS: 65534
Age: 4:48 Metric2: 20
Task: BGP_65534.10.1.2.3+2244
Announcement bits (2): 2-KRT 5-BGP_Sync_Any
AS path: 1 10 I
BGP next hop: 192.168.129.50
Localpref: 100
Router ID: 10.1.2.3
```

user@Router 4> show route 1/8 detail

inet.0: 275 destinations, 275 routes (274 active, 0 holddown, 1 hidden)

+ = Active Route, - = Last Active, * = Both

1.0.0.0/8 (1 entry, 1 announced)

```
*BGP Preference: 170/-101
Nexthop: 192.168.129.194 via so-4/0/0.0, selected
State: <Active Ext>
Local AS: 6 Peer AS: 65534
Age: 5:09
Task: BGP_65534.192.168.129.194+179
Announcement bits (2): 2-KRT 6-BGP0.0.0.0+179
AS path: 65534 1 10 I
Localpref: 100
Router ID: 10.1.2.3
```

Remove Private AS Numbers from AS Paths

By default, when BGP advertises AS paths to remote systems, it includes all AS numbers, including private AS numbers. You can configure the software so that it removes private AS numbers from AS paths. Doing this is useful when all the following circumstances are true:

- A remote AS for which you provide connectivity is multihomed, but only to the local AS.

- The remote AS does not have an officially allocated AS number.

- It is not appropriate to make the remote AS a confederation member AS of the local AS.

To have the local system strip private AS numbers from the AS path, include the `remove-private` statement:

```
remove-private;
```

You can remove private AS numbers for all BGP groups (at the `[edit protocols bgp]` hierarchy level) or for all peers in a group (at the `[edit protocols bgp group group-name]` level).

The AS numbers are stripped from the AS path starting at the left end of the AS path (the end where AS paths have been most recently added). This operation takes place after any confederation member ASs have already been removed from the AS path, if applicable.

The software is preconfigured with knowledge of the set of AS numbers that is considered private, a range that is defined in the IANA assigned numbers document. The set of AS numbers reserved as private are in the range 64512 through 65534, inclusive.

Configure Route Reflection

In standard internal BGP implementations, all BGP systems within the AS are fully meshed so that any external routing information is redistributed among all routers within the AS. This type of implementation can present scaling issues when an AS has a large number of internal BGP systems because of the amount of identical information that BGP systems must share with each other. Route reflection provides one means of decreasing BGP control traffic, minimizing the number of update messages sent within the AS.

In route reflection, BGP systems are arranged in *clusters*. Each cluster consists of at least one system that acts as a *route reflector*, along with any number of *client peers*. BGP peers outside the cluster are called *nonclient peers*. The route reflector reflects (redistributes) routing information to each client peer (*intracluster reflection*) and to all nonclient peers (*intercluster reflection*). Because the route reflector redistributes routes within the cluster, the BGP systems within the cluster do not have to be fully meshed.

When the route reflector receives a route, it selects the best path. Then, if the route came from a nonclient peer, the route reflector sends the route to all client peers within the cluster. If the route came from a client peer, the route reflector sends it to all nonclient peers and to all client peers except the originator. In this process, none of the client peers sends routes to other client peers.

To configure route reflection, you specify a cluster identifier only on the BGP systems that are to be the route reflectors. These systems then determine, from the network reachability information they receive, which BGP systems are part of its cluster and are client peers, and which BGP systems are outside the cluster and are nonclient peers.

To configure a router to be a route reflector, you must do the following:

- Configure multiple IBGP groups.

- Configure a cluster identifier (using the `cluster` statement) for groups that are members of the cluster.

- Configure all the groups with the same IBGP AS number.

To configure the route reflector, include the following statements in the configuration:

```
[edit protocols bgp]
group group-name {
  type internal;           # Peer group consisting of the nonclient peers of the local system,
                          # which is acting as a route reflector
  peer-as autonomous-system;
  neighbor address1;
  neighbor address2;
}
group group-name {
  type internal;           # Peer group consisting of the peers that are in the cluster;
                          # each peer is treated as a route reflector client by the local system
  peer-as autonomous-system;      # Number must match previous peer-as statement
  cluster cluster-identifier;
  neighbor address3;
  neighbor address4;
}
```

By default, the BGP route reflector performs intracluster reflection because it assumes that all the client peers are not fully meshed. However, if the client peers are fully meshed, intracluster reflection results in the sending of redundant route advertisements. In this case, you can disable intracluster reflection by including the `no-client-reflect` statement within the group statement:

```
group group-name {
  type internal;           # Peer group consisting of the peers that are in the cluster;
                          # each peer is treated as a route reflector client by the local system
  peer-as autonomous-system;      # Number must match previous peer-as statement
  cluster cluster-identifier;
  no-client-reflect;
  neighbor address3;
  neighbor address4;
}
```

Examples: Configure Route Reflection

This example shows how to configure a simple route reflector. The configuration contains three routes: Router 1, which is the route reflector; Router 2, which is a client; and Router 3, which is a nonclient.

The routers have the following loopback addresses:

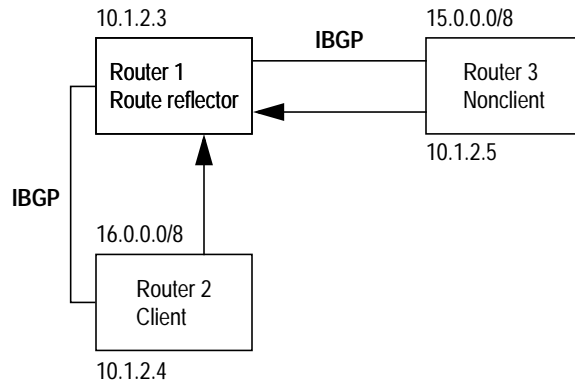
Router 1—10.1.2.3

Router 2—10.1.2.4

Router 3—10.1.2.5

You must configure all routers to run a common IGP or to have static configuration, so that they learn each other's loopback addresses.

Figure 11: Simple Route Reflector



Configure Router 1 to be a route reflector for Router 2 and a regular IBGP neighbor for Router 3:

```

[edit]
routing-options {
  autonomous-System 65534;
}
protocols {
  bgp {
    group 13 {
      type internal;
      local-address 10.1.2.3;
      neighbor 10.1.2.5;
    }
    group 12 {
      type internal;
      local-address 10.1.2.3;
      cluster 1.2.3.4;
      neighbor 10.1.2.4;
    }
  }
}
  
```

Configure Router 2 to be an IBGP neighbor to Router 1 and announce 16.0.0.0/8 to Router 1.
Configure route 16.0.0.0/8 as a static route on Router 2.

```
[edit]
routing-options {
  static {
    route 16.0.0.0/8 nexthop 172.16.1.2;
  }
  autonomous-system 65534;
}
protocols {
  bgp {
    group 21 {
      type internal;
      local-address 10.1.2.4;
      export dist-static;
      neighbor 10.1.2.3;
    }
  }
}
policy-options {
  policy-statement dist-static {
    from protocol static;
    then accept;
  }
}
```

Configure Router 3 to be an IBGP neighbor to Router 1 and announce 15.0.0.0/8 to Router 1.
Configure route 15.0.0.0/8 as a static route on Router 3.

```
[edit]
routing-options {
  static {
    route 15.0.0.0/8 nexthop 172.16.1.2;
  }
  autonomous-system 65534;
}
protocols {
  bgp {
    group 31 {
      type internal;
      local-address 10.1.2.5;
      export dist-static;
      neighbor 10.1.2.3;
    }
  }
}
policy-options {
  policy-statement dist-static {
    from protocol static;
    then accept;
  }
}
```

The following is the output of the show route detail command for route 16.0.0.0/8 on Router 1 and Router 3. Note that Router 1 learns 16.0.0.0/8 from its client, Router 2, and reflects it to Router 3. On Router 3, the output of the show route commands include the cluster list and originator ID attributes, which are added by Router 1 when the route is reflected.

Router 1:

```
user@router1> show route 16.0.0.0/8 detail
```

```
inet.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
16.0.0.0/8 (1 entry, 1 announced)
```

```
*BGP Preference: 170/-101
Source: 10.1.2.4
Nexthop: 172.16.1.2 via fxp0.0, selected
State: <Active Int Ext>
Local AS: 65534 Peer AS: 65534
Age: 11:55 Metric2: 0
Task: BGP_65534.10.1.2.4+4327
Announcement bits (3): 2-KRT 3-BGP0.0.0.0+179 4-BGP_Sync_Any
AS path: I
BGP next hop: 172.16.1.2
Localpref: 100
Router ID: 10.1.2.4
```

Router 3:

```
user@router3> show route 16.0.0.0/8 detail
```

```
inet.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
16.0.0.0/8 (1 entry, 1 announced)
```

```
*BGP Preference: 170/-101
Source: 10.1.2.3
Nexthop: 172.16.1.2 via fxp0.0, selected
State: <Active Int Ext>
Local AS: 65534 Peer AS: 65534
Age: 11:57 Metric2: 0
Task: BGP_65534.10.1.2.3+4619
Announcement bits (2): 2-KRT 4-BGP_Sync_Any
AS path: I <Originator>
Cluster list: 1.2.3.4
Originator ID: 10.1.2.4
BGP next hop: 172.16.1.2
Localpref: 100
Router ID: 10.1.2.3
```

The following is the output of the show route detail command for route 15.0.0.0/8 on Router 1 and Router 2. Similar to when routes are reflected from client peers to nonclient peers, Router 1 reflects a route it learns from a regular IBGP neighbor to its client. Cluster list and Originator ID attributes are added during the reflection process.

Router 1:

```
user@router1> show route 15.0.0.0/8 detail
```

```
inet.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
15.0.0.0/8 (1 entry, 1 announced)
```

```
*BGP Preference: 170/-101
Source: 10.1.2.5
Nexthop: 172.16.1.2 via fxp0.0, selected
State: <Active Int Ext>
Local AS: 65534 Peer AS: 65534
Age: 11:14 Metric2: 0
Task: BGP_65534.10.1.2.5+179
Announcement bits (3): 2-KRT 3-BGP0.0.0.0+179 4-BGP_Sync_Any
AS path: I
BGP next hop: 172.16.1.2
Localpref: 100
Router ID: 10.1.2.5
```

```
user@router2> show route 15.0.0.0/8 detail
```

```
inet.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
15.0.0.0/8 (1 entry, 1 announced)
```

```
*BGP Preference: 170/-101
Source: 10.1.2.3
Nexthop: 172.16.1.2 via fxp0.0, selected
State: <Active Int Ext>
Local AS: 65534 Peer AS: 65534
Age: 11:23 Metric2: 0
Task: BGP_65534.10.1.2.3+179
Announcement bits (2): 2-KRT 4-BGP_Sync_Any
AS path: I <Originator>
Cluster list: 1.2.3.4
Originator ID: 10.1.2.5
BGP next hop: 172.16.1.2
Localpref: 100
Router ID: 10.1.2.3
```

Enable Route Flap Damping

BGP *route flapping* describes the situation in which BGP systems send an excessive number of update messages to advertise network reachability information. BGP *flap damping* is a method of reducing the number of update messages sent between BGP peers, thereby reducing the load on these peers, without adversely affecting the route convergence time for stable routes.

By default, route flap damping is disabled. To enable it, include the damping statement:

```
damping;
```

You can configure route flap damping for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

Damping is applied to external peers and to peers at confederation boundaries. For finer control over which peers have damping enabled, include the damping statement at the [edit protocols bgp group *group-name*] level.

By default, route flap damping uses the following parameters:

Decay half-life while reachable—15 minutes

Maximum hold-down time—60 minutes

Reuse threshold—750

Cut-off threshold—3000

To change these default parameters, you must define the flap damping parameters with the damping statement at the [edit policy-options] hierarchy level and then apply them using an import statement when configuring BGP. For more information about flap damping and defining flap damping parameters, see “Configure Damping Parameters” on page 77. For more information about applying policy filters in BGP, see “Configure BGP Routing Policy” on page 336.

Enable Multiprotocol BGP

Multiprotocol BGP (MBGP) is an extension to BGP that enables BGP to carry routing information for multiple network layers and address families. MBGP can carry the unicast routes used for multicast routing separately from the routes used for unicast IP forwarding.

To enable MBGP, you configure BGP to carry network layer reachability information (NLRI) for address families other than unicast IPv4 by including the family inet statement:

```
family inet {
  (any | multicast | unicast) {
    prefix-limit {
      maximum number;
      teardown <percentage>;
    }
    rib-group routing-table-name;
  }
}
```

You can enable MBGP for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

By default, BGP peers carry only unicast routes used for unicast forwarding purposes. To configure BGP peers to carry only multicast routes, specify the multicast option. To configure BGP peers to carry both unicast and multicast routes, specify the any option.

When MBGP is configured, BGP installs the MBGP routes into different routing tables. Each routing table is identified by the protocol family or address family indicator (AFI) and a subaddress family indicator (SAFI). The JUNOS software supports AFI inet and SAFI 1 for unicast routes, AFI inet and SAFI 2 for multicast sources, and AFI inet and SAFI 3 for both unicast and multicast prefixes. If BGP receives a prefix (NLRI) with SAFI 1, it places this route into inet.0. It places NLRIs with SAFI 2 into inet.2 and those with SAFI 3 into both inet.0 and inet.2.

When you enable multiprotocol BGP, you can do the following:

- Limit the Number of Prefixes on a BGP Peering on page 335

- Configure BGP Routing Table Groups on page 335

Limit the Number of Prefixes on a BGP Peering

You can limit the number of prefixes received on a BGP peering and rate-limit logging when the number of injected prefixes exceeds a set limit. You can also tear down the peering when the number of prefixes exceeds the limit.

To configure a maximum number of prefixes, include the `prefix-limit` statement:

```
prefix-limit {
  maximum number;
  teardown <percentage>;
}
```

You can specify the maximum number of prefixes at the [edit protocols bgp family inet (any | multicast | unicast)] hierarchy level, the [edit protocols bgp group *group-name* family inet (any | multicast | unicast)] level, or the [edit protocols bgp group *group-name* neighbor *address* family inet (any | multicast | unicast)] level. For routing instances, include the statement at the [edit routing-instances *routing-instance-name* protocols bgp family inet (any | multicast | unicast)] hierarchy level or the [edit routing-instances *routing-instance-name* protocols bgp group *group-name* neighbor *address* family inet (any | multicast | unicast)] hierarchy level.

When you set the maximum number of prefixes, a message is logged when that number is reached.

If you include the `teardown` statement, the session is torn down when the maximum number of prefixes is reached. If you specify a percentage, messages are logged when the number of prefixes reaches that percentage of the maximum.

Configure BGP Routing Table Groups

When a BGP session receives a unicast or multicast NLRI, it installs the route in the appropriate table (inet.0 for unicast and inet.2 for multicast). To add unicast prefixes to unicast and multicast tables, you can configure BGP routing table groups. This is useful if you cannot perform multicast NLRI negotiation.

To configure BGP routing table groups, include the `rib-group` statement:

```
family inet {
  (any | multicast | unicast) {
    rib-group routing-table-name;
  }
}
```

You can include this statement at the [edit protocols bgp family inet] hierarchy level, the [edit protocols bgp group *group-name* family inet] level, or the [edit protocols bgp group *group-name* neighbor *address* family inet] level.

Configure BGP Routing Policy

All routing protocols use the JUNOS software routing table to store the routes that they learn and to determine which routes they should advertise in their protocol packets. Routing policy allows you to control which routes the routing protocols store in and retrieve from the routing table. For information about routing policy, see “Configure Routing Policy” on page 35.

When configuring BGP routing policy, you can perform the following tasks:

Apply Routing Policy on page 336

Have BGP Advertise Nonactive Routes on page 339

Configure How Often BGP Exchanges Routes with the Routing Table on page 339

Apply Routing Policy

You define routing policy at the [edit policy-options] hierarchy level. To apply policies you have defined for BGP, include the `import` and `export` statements within the BGP configuration. For information about defining policy, see “Configure Routing Policy” on page 35.

You can apply policies as follows:

BGP global `import` and `export` statements—Include these statements at the [edit protocols bgp] hierarchy level.

Group `import` and `export` statements—Include these statements at the [edit protocols bgp group *group-name*] hierarchy level.

Peer `import` statements—Include these statements at the [edit protocols bgp group *group-name* neighbor *address*] hierarchy level. (A peer uses its group’s `export` statement.)

A peer-level `import` statement overrides a group `import` statement. A group-level `import` or `export` statement overrides a global BGP `import` or `export` statement.

Apply Policies to Routes Being Imported into the Routing Table from BGP

To apply policy to routes being imported into the routing table from BGP, include the import statement, listing the names of one or more policies to be evaluated:

```
import [policy-names];
```

If you specify more than one policy, they are evaluated in the order specified, from first to last, and the first matching filter is applied to the route. If no match is found, BGP places into the routing table only those routes that were learned from BGP routers.

Apply Policies to Routes Being Exported from the Routing Table into BGP

To apply policy to routes being exported from the routing table into BGP, include the export statement, listing the names of one or more policies to be evaluated:

```
export [policy-names];
```

If you specify more than one policy, they are evaluated in the order specified, from first to last, and the first matching filter is applied to the route. If no routes match the filters, the routing table exports into BGP only the routes that it learned from BGP.

BGP-Specific Policy Filter Match Conditions and Actions

The following policy match conditions are specific for BGP. For more information about the match conditions, see Table 2 on page 38.

AS path regular expressions—To define AS path regular expressions for matching routes, include the `as-path` statement at the [edit policy-options] hierarchy level, then include the `as-path` match condition in the `from` or `to` clause of the policy.

Communities—A BGP *community* is a group of prefixes that share a common property. Community information is included as a path attribute in BGP update messages. You can define routing policy matches based on communities. To define the community, include the `community` statement at the [edit policy-options] hierarchy level, then include the community match conditions in the `from` or `to` clause of the policy.

Local preference—To define the local preference for matching routes, include the `local-preference` match condition in the `from` or `to` clause of the policy.

Multiple exit discriminator (MED)—To define the MED for matching routes, include the `metric` match condition in the `from` or `to` clause of the policy.

Origin—To define the path origin for matching routes, include the `origin` match condition in the `from` or `to` clause of the policy.

The following policy actions are specific for BGP. For more information about the actions, see Table 4 on page 41.

AS paths—To place one or more AS numbers at the beginning of the AS path, include the `as-path-prepend` action in the `then` clause of the policy.

Communities—You can control the communities included in routes. To define the community, include the `community` statement at the `[edit policy-options]` hierarchy level, then include the `community add`, `community delete`, or `community set` action in the `then` clause of the policy.

Route flap damping—BGP route flap damping is a method of reducing the number of update messages sent between BGP peers. You can define policy actions that affect flap damping. To define damping policies, include the `damping` statement at the `[edit policy-options]` hierarchy level, then include the `damping` action in the `then` clause of the policy.

Local preference—To define the local preference for matching routes, include the `local-preference` action in the `then` clause of the policy.

Multiple exit discriminator (MED)—To define the MED for matching routes, include the `metric` action in the `then` clause of the policy.

Origin—To define the path origin for matching routes, include the `origin` action in the `then` clause of the policy.

Example: Apply Routing Policy Filters

Define a policy filter that changes the metric for all routes with a particular AS path, and apply this policy filter to the routes that the routing table imports from a BGP group:

```
[edit]
routing-options {
  autonomous-system 56;
}
policy-options {
  as-path auckland "23 666 (777 | 888) .*";
  policy-statement change-metric {
    from as-path auckland;
    then metric 20;
  }
}
protocols {
  bgp {
    group 23 {
      type external;
      peer-as 23;
      import change-metric;
      neighbor 10.0.0.1;
      neighbor 10.0.0.3;
    }
  }
}
```

Have BGP Advertise Nonactive Routes

By default, BGP stores the route information it receives from update messages in the JUNOS routing table, and the routing table exports only active routes into BGP, which BGP then advertises to its peers. To have the routing table export to BGP all routes learned by BGP even if the JUNOS software did not select them to be active routes, include the `advertise-inactive` statement:

```
advertise-inactive;
```

You can advertise nonactive routes for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

Configure How Often BGP Exchanges Routes with the Routing Table

BGP stores the route information it receives from update messages in the routing table, and the routing table exports active routes from the routing table into BGP. BGP then advertises the exported routes to its peers. By default, the exchange of route information between BGP and the routing table occurs immediately after the routes are received. This immediate exchange of route information might cause instabilities in the network reachability information. To guard against this, you can delay the time between when BGP and the routing table exchange route information.

To configure a delay between when the routing table receives route information and when it exports that information to BGP, include the `out-delay` statement:

```
out-delay seconds;
```

You can configure the delay for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

By default, the routing table retains some of the route information learned from BGP. To have the routing table retain all or none of this information, include the `keep` statement:

```
keep (all | none);
```

You can configure the amount of information retained for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

The routing table can retain the route information learned from BGP in one of the following ways:

Default (omit the `keep` statement)—Keep all route information that was learned from BGP except for routes whose AS path is looped and the loop includes the local AS.

`keep all`—Keep all route information that was learned from BGP.

`keep none`—Discard routes that were received from a peer and that were rejected by import policy or other sanity checking, such as AS path or next hop.

In an AS path healing situation, routes with looped paths theoretically could become usable during a soft reconfiguration when the AS path loop limit is changed. However, there is a significant memory usage difference between the default and keep all because it is common for a peer to readvertise routes back to the peer from which it learned them. The default behavior is not to waste memory on such routes.

Refresh BGP Routes

To dynamically request readvertisement of routes from a peer, you can refresh BGP routes. When keep none is configured for the BGP session and the inbound policy changes, the JUNOS software forces readvertisement of the full set of routes advertised by the peer. To perform BGP route refresh, run the following command:

```
clear bgp neighbor address soft-inbound;
```

You can refresh routes for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).



Note

When a router supports BGP route refresh, the show bgp neighbor command displays *Peer supports Refresh capability* near the NLRI section. For more information about show commands, see the *JUNOS Internet Software Command Reference*.

Configure BGP to Log Syslog Messages

Whenever a BGP peer makes a state transition, you can configure BGP so that it generates a syslog message. To do this, include the log-updown statement:

```
log-updown;
```

You can configure the generation of syslog messages for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

Describe BGP Router Configuration

You can enter plain text to describe the BGP router configuration.

To enter a description, include the description statement:

```
description description-text;
```

You can enter a description for all BGP groups (at the [edit protocols bgp] hierarchy level), for all peers in a group (at the [edit protocols bgp group *group-name*] level), or for an individual peer (at the [edit protocols bgp group *group-name* neighbor *address*] level).

Trace BGP Protocol Traffic

To trace BGP protocol traffic, you can specify options in the global traceoptions statement at the [edit routing-options] hierarchy level, and you can specify BGP-specific options by including the traceoptions statement at the [edit protocols bgp] hierarchy level:

```
[edit protocols bgp]
traceoptions {
  file name <replace> <size size> <files number> <no-stamp>
    <(world-readable | no-world-readable)>;
  flag flag <flag-modifier> <disable>;
}
```

You can specify the following BGP-specific options in the BGP traceoptions statement:

aspath—Trace AS path regular expression operations.

damping—Trace damping operations.

keepalive—Trace BGP keepalive messages.

open—Trace BGP open packets. These packets are sent between peers when they are establishing a connection.

packets—Trace all BGP protocol packets.

update—Trace update packets. These packets provide routing updates to BGP systems.

For general information about tracing, see tracing and logging information in the *JUNOS Internet Software Configuration Guide: Installation and System Management*.

Examples: Trace BGP Protocol Traffic

Trace only unusual or abnormal operations to routing-log, and trace detailed information about all BGP messages to bgp-log:

```
[edit]
routing-options {
  traceoptions {
    file routing-log;
  }
  autonomous-system 23;
}
protocols{
  bgp {
    group 23 {
      type external;
      peer-as 56;
      traceoptions {
        file bgp-log size 10k files 5;
        flag packets detail;
      }
      allow 0.0.0.0/0;
    }
  }
}
```

Trace only update messages received from the configured peer:

```
[edit]
routing-options{
  autonomous-system 23;
  router-id 10.0.0.1;
}
protocols {
  bgp {
    group 23 {
      type external;
      peer-as 56;
      neighbor boojum.snark.net {
        traceoptions {
          file bgp-log size 10k files 2;
          flag update detail;
        }
      }
    }
  }
}
```