Engineering
Simplicity

Junos® OS

Protocol-Independent Routing Properties User Guide

Published
2025-12-15

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

YEAR 2000 NOTICE

END USER LICENSE AGREEMENT

Table of Contents

About This Guide

Use this guide to configure, monitor, and troubleshoot protocol-independent routing properties on your Juniper Network devices.

1

CHAPTER

Overview

IN THIS CHAPTER

Protocol-Independent Routing Properties Overview

In Junos OS, routing capabilities and features that are not specific to any particular routing protocol are collectively called protocol-independent routing properties. These features often interact with routing protocols. In many cases, you combine protocol-independent properties and routing policy to achieve a goal. For example, you define a static route using protocol-independent properties, and then, using a routing policy, you can redistribute the static route into a routing protocol, such as BGP, OSPF, or IS-IS.

Protocol-independent routing properties include:

- Static, aggregate, and generated routes
- Bidirectional Forwarding Detection on static routes
- Global preference
- Martian routes
- Routing tables and routing information base (RIB) groups

2

CHAPTER

Junos OS Routing Tables

IN THIS CHAPTER

Configuring Junos OS Routing Tables

IN THIS SECTION

Understanding Junos OS Routing Tables

Junos OS automatically creates and maintains several routing tables. Each routing table is used for a specific purpose. In addition to these automatically created routing tables, you can create your own routing tables.

Each routing table populates a portion of the forwarding table. Thus, the forwarding table is partitioned based on routing tables. This allows for specific forwarding behavior for each routing table. For example, for VPNs, each VPN-based routing table has its own VPN-specific partition in the forwarding table.

It is common for the routing software to maintain unicast routes and multicast routes in different routing tables. You also might have policy considerations that would lead you to create separate routing tables to manage the propagation of routing information.

Creating routing tables is optional. If you do not create any, Junos OS uses its default routing tables, which are as follows:

- **inet.0**—For IP version 4 (IPv4) unicast routes. This table stores interface local and direct routes, static routes, and dynamically learned routes.
- **inet.1**—For the IPv4 multicast forwarding cache. This table stores the IPv4 (S,G) group entries that are dynamically created as a result of join state information.
- **inet.2**—For subsequent address family indicator (SAFI) 2 routes, when multiprotocol BGP (MBGP) is enabled. This table stores unicast routes that are used for multicast reverse-path-forwarding (RPF) lookup. The routes in this table can be used by the Distance Vector Multicast Routing Protocol (DVMRP), which requires a specific RPF table. In contrast, Protocol Independent Multicast (PIM) does not need this table because it can perform RPF checks against the inet.0 table. You can import routes

from inet.0 into inet.2 using routing information base (RIB) groups, or install routes directly into inet.2 from a multicast routing protocol.

- **inet.3**—For IPv4 MPLS. This table stores the egress address of an MPLS label-swiched path (LSP), the LSP name, and the outgoing interface name. This routing table is used only when the local device is the ingress node to an LSP.
- **inet6.0**—For IP version 6 (IPv6) unicast routes. This table stores interface local and direct routes, static routes, and dynamically learned routes.
- **inet6.1**—For IPv6 multicast forwarding cache. This table stores the IPv6 (S,G) group entries that are dynamically created as a result of join state information.
- **inet6.2**—The inet6.2 table is often used in conjunction with other IPv6 routing tables as part of the default routing table groups for interface routes, particularly on PTX routers.
- **inet6.3**—The inet6.3 table is used for storing labeled IPv6 routes.
- ***instance-name*.inet.0**—If you configure a routing instance, Junos OS creates the default unicast routing table ***instance-name*.inet.0**.
- ***instance-name*.inet.2**—If you configure **routing-instances *instance-name* protocols bgp family inet multicast** in a routing instance of type VRF, Junos OS creates the ***instance-name*.inet.2** table.

  Another way to create the ***instance-name*.inet.2** table is to use the `rib-group` statement. See "Example: Exporting Specific Routes from One Routing Table Into Another Routing Table" on page 14.

  NOTE: Importing **inet-vpn multicast** routes from the **bgp.l3vpn.2** table into the ***instance-name*.inet.2** table does not create the ***instance-name*.inet.2** table. The import operation works only if the ***instance-name*.inet.2** table already exists.

- ***instance-name*.inetflow.0**—If you configure a flow route, Junos OS creates the flow routing table ***instance-name*.inetflow.0**.
- **bgp.l2vpn.0**—For Layer 2 VPN routes learned from BGP. This table stores routes learned from other provider edge (PE) routers. The Layer 2 routing information is copied into Layer 2 VPN routing and forwarding instances (VRFs) based on target communities.
- **bgp.l3vpn.0**—For Layer 3 VPN routes learned from BGP. This table stores routes learned from other PE routers. Routes in this table are copied into a Layer 3 VRF when there is a matching route table.
- `l2circuit.0`—For l2circuit routes learned from LDP. Routes in this table are used to send or receive l2circuit signaling messages.
- **mpls.0**—For MPLS label switching operations. This table is used when the local device is a transit router.

- **iso.0**—For IS-IS routes. When you are using IS-IS to support IP routing, this table contains only the local device's network entity title (NET).
- **juniper_private**—For Junos OS to communicate internally between the Routing Engine and PIC hardware.
- **l2xc.0**—The table holds Layer 2 cross-connect routes for CCC and related Layer 2 switching services in Junos OS over a SRv6 network.
- **lsdist.0**—The table in Junos OS holds BGP-LS NLRIs, serving as the key integration point between the traffic engineering database and BGP-LS route advertisement and reception.
- **vxlan.inet**—For VXLAN tunnelling operations on certain QFX platforms. This table stores reachability to VXLAN tunnel endpoints over an IP network. It is used for VXLAN encapsulation and decapsulation.
- **bgp.evpn0**—For BGP EVPN routes. This table stores routes learned through BGP for EVPN operations, including MAC and IP address bindings, inclusive multicast routes, and Ethernet segment routes.
- **instance-name.evpn**—For EVPN routes in a specific routing instance. When you configure an EVPN routing instance, Junos and EVO OS creates this table to store the EVPN routes specific to that instance.

Routing Table Features in Junos OS

Junos OS maintains two databases for routing information:

- Routing table—Contains all the routing information learned by all routing protocols. (Some vendors refer to this kind of table as a routing information base [RIB].)
- Forwarding table—Contains the routes actually used to forward packets. (Some vendors refer to this kind of table as a forwarding information base [FIB].)

By default, Junos OS maintains three routing tables: one for IP version 4 (IPv4) unicast routes, a second for multicast routes, and a third for MPLS. You can configure additional routing tables.

The Junos OS maintains separate routing tables for IPv4 and IP version 6 (IPv6) routes.

The Junos OS installs all active routes from the routing table into the forwarding table. The active routes are routes that are used to forward packets to their destinations. The Junos operating system kernel maintains a master copy of the forwarding table. It copies the forwarding table to the Packet Forwarding Engine, which is the component responsible for forwarding packets.

The Junos routing protocol process generally determines the active route by selecting the route with the lowest preference value. The Junos OS provides support for alternate and tiebreaker preferences, and some of the routing protocols, including BGP and MPLS, use these additional preferences.

You can add martian addresses and static, aggregate, and generated routes to the Junos routing tables, configuring the routes with one or more of the properties shown in Table 1 on page 7.

Table 1: Routing Table Route Properties

Description	Static	Aggregate	Generated
Destination address	X	X	X
Default route to the destination	X	X	X
IP address or interface of the next hop to the destination	X	–	–
Label-switched path (LSP) as next hop	X	–	–
Drop the packets, install a reject route for this destination, and send Internet Control Message Protocol (ICMP) unreachable messages	X	X	X
Drop the packets, install a reject route for this destination, but do not send ICMP unreachable messages	X	X	X
Cause packets to be received by the local router	X	–	–
Associate a metric value with the route	X	X	X
Type of route	X	X	X
Preference values	X	X	X
Additional preference values	X	X	X
Independent preference (**qualified-next-hop** statement)	X	–	–

Table 1: Routing Table Route Properties *(Continued)*

Description	Static	Aggregate	Generated
BGP community information to associate with the route	X	X	X
Autonomous system (AS) path information to associate with the route	X	X	X
OSPF tag strings to associate with the route	X	X	X
Do not install active static routes into the forwarding table	X	–	–
Install the route into the forwarding table	X	–	–
Permanently retain a static route in the forwarding table	X	–	–
Include only the longest common leading sequences from the contributing AS paths	–	X	–
Include all AS numbers for a specific route	–	X	–
Retain an inactive route in the routing and forwarding tables	X	X	X
Remove an inactive route from the routing and forwarding tables	X	X	X
Active policy to associate with the route	–	X	X
Specify that a route is ineligible for readvertisement	X	–	–
Specify route to a prefix that is not a directly connected next hop	X	–	–

Understanding Default Routing Table Groups for Interface Routes on PTX Routers

On PTX Series Packet Transport Routers, the default interface-route routing table groups differ from that of other Junos OS routing devices.

The PTX Series routers are MPLS transit platforms that do IP forwarding, typically using interior gateway protocol (IGP) routes. Interface routes are directly connected and local routes.

PTX Series routers are unlike other Junos OS routing devices in that they force an indirect next-hop resolution. PTX Series routers need the indirect next hop be resolved to create the chained composite next hop. This can cause routes to be hidden when the next-hop type is unusable.

To prevent routes from being hidden, PTX Series platforms automatically copy the routes in inet.0 into inet.2 and inet.3, and the routes in inet6.0 into inet6.2 and inet6.3.

The default interface routing table configuration on the PTX Series routers is as follows:

```
user@host# show routing-options | display inheritance defaults
##
## 'interface-routes' was inherited from group 'junos-defaults'
##
interface-routes {
    ##
    ## 'rib-group' was inherited from group 'junos-defaults'
    ##
    rib-group {
        ##
        ## 'junos-ifrg-inet0-to-inet2-and-inet3' was inherited from group 'junos-defaults'
        ##
        inet junos-ifrg-inet0-to-inet2-and-inet3;
        ##
        ## 'junos-ifrg-inet60-to-inet62-and-inet63' was inherited from group 'junos-defaults'
        ##
        inet6 junos-ifrg-inet60-to-inet62-and-inet63;
    }
}
rib-groups {
    ##
    ## 'junos-ifrg-inet0-to-inet2-and-inet3' was inherited from group 'junos-defaults'
    ##
    junos-ifrg-inet0-to-inet2-and-inet3 {
        ##
        ## 'inet.0' was inherited from group 'junos-defaults'
        ## 'inet.2' was inherited from group 'junos-defaults'
        ## 'inet.3' was inherited from group 'junos-defaults'
        ##
        import-rib [ inet.0 inet.2 inet.3 ];
    }
    ##
    ## 'junos-ifrg-inet60-to-inet62-and-inet63' was inherited from group 'junos-defaults'
    ##
    junos-ifrg-inet60-to-inet62-and-inet63 {
        ##
        ## 'inet6.0' was inherited from group 'junos-defaults'
        ## 'inet6.2' was inherited from group 'junos-defaults'
        ## 'inet6.3' was inherited from group 'junos-defaults'
        ##
        import-rib [ inet6.0 inet6.2 inet6.3 ];
    }
}
```

SEE ALSO

Example: Overriding the Default BGP Routing Policy on PTX Series Packet Transport Routers

Example: Creating Routing Tables

IN THIS SECTION

This example shows how to create a custom routing table.

Requirements

In this example, no special configuration beyond device initialization is required.

Overview

Creating routing tables is optional. You might have policy considerations that would lead you to create separate routing tables to manage the propagation of routing information. This capability is rarely used, but it is demonstrated here for completeness.

If you do not create any routing tables, Junos OS uses its default routing tables.

NOTE: If you want to add static, aggregate, generated, or martian routes only to the default IPv4 unicast routing table (**inet.0**), you do not have to create any routing tables because, by default, these routes are added to **inet.0**. You can add these routes by including the **static**, **aggregate**, **generate**, and `martians` statements.

To explicitly create a routing table, include the `rib` statement and child statements under the `rib` statement.

The routing table name, ***routing-table-name***, includes the protocol family, optionally followed by a period and a number. The protocol family can be **inet** for the IPv4 family, **inet6** for the IPv6 family, or **iso** for the International Standards Organization (ISO) protocol family. The number represents the routing instance. The first instance is 0.

This example shows how to configure a custom IPv4 routing table called inet.14. The example also shows how to populate the routing table with a single static route.

NOTE: On EX Series switches, only dynamically learned routes can be imported from one routing table group to another.

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

```
set routing-options rib inet.14 static route 10.2.0.0/16 discard
```

Procedure

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To create a routing table:

1. Configure the routing table.

   ```
   [edit routing-options]
   user@host# set rib inet.14 static route 10.2.0.0/16 discard
   ```

2. If you are done configuring the device, commit the configuration.

   ```
   [edit]
   user@host# commit
   ```

Results

Confirm your configuration by issuing the `show routing-options` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show routing-options
rib inet.14 {
    static {
        route 10.2.0.0/16 discard;
```

```
    }
}
```

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Checking the Routing Table

Purpose

Make sure that the static route appears in the custom routing table.

Action

```
user@host> show route table inet.14
inet.14: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
Restart Complete
+ = Active Route, - = Last Active, * = Both

10.2.0.0/16        *[Static/5] 00:00:09
                      Discard
```

Meaning

The static route is in the custom routing table.

Example: Exporting Specific Routes from One Routing Table Into Another Routing Table

IN THIS SECTION

This example shows how to duplicate specific routes from one routing table into another routing table within the same routing instance.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

This example uses the `auto-export` statement and the `rib-group` statement to accomplish the goal of exporting specific routes from one routing table to another.

Consider the following points:

- When **auto-export** is configured in a routing instance, the **vrf-import** and **vrf-export** policies are examined. Based on the route target and community information in the policies, the **auto-export** function performs route leaking among the local routing instance inet.0 tables.
- You can use the `rib-group` statement if it is necessary to import routes into tables other than *instance*.inet.0. To use a RIB group with **auto-export**, the routing instance should specify explicit **vrf-import** and **vrf-export** policies. The **vrf-import** and **vrf-export** policies can be extended to contain additional terms to filter routes as needed for the RIB group.

In this example, access-internal routes are added into the vpna.inet.0 routing table. The access-internal routes are also duplicated into the vpna.inet.2 routing table.

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

```
set interfaces fe-1/3/1 vlan-tagging
set interfaces fe-1/3/1 unit 0 vlan-id 512
set interfaces fe-1/3/1 unit 0 family inet address 10.168.100.3/24
set interfaces lo0 unit 0 family inet address 192.168.3.3/32
set routing-options rib-groups rib-group-vpna-access-internal import-rib vpna.inet.2
set routing-options autonomous-system 63000
set policy-options policy-statement vpna-export term a from protocol bgp
set policy-options policy-statement vpna-export term a then community add vpna-comm
set policy-options policy-statement vpna-export term a then accept
set policy-options policy-statement vpna-export term b from protocol access-internal
set policy-options policy-statement vpna-export term b then accept
set policy-options policy-statement vpna-export term c then reject
set policy-options policy-statement vpna-import term a from protocol bgp
set policy-options policy-statement vpna-import term a from community vpna-comm
set policy-options policy-statement vpna-import term a then accept
set policy-options policy-statement vpna-import term b from instance vpna
set policy-options policy-statement vpna-import term b from protocol access-internal
set policy-options policy-statement vpna-import term b then accept
set policy-options policy-statement vpna-import term c then reject
set policy-options community vpna-comm members target:63000:100
set routing-instances vpna instance-type vrf
set routing-instances vpna interface fe-1/3/1.1
set routing-instances vpna route-distinguisher 100:1
set routing-instances vpna vrf-import vpna-import
set routing-instances vpna vrf-export vpna-export
set routing-instances vpna routing-options auto-export family inet unicast rib-group rib-group-
```

```
vpna-access-internal
set routing-instances vpna protocols bgp group bgp-vpna type external
set routing-instances vpna protocols bgp group bgp-vpna family inet multicast
set routing-instances vpna protocols bgp group bgp-vpna peer-as 100
set routing-instances vpna protocols bgp group bgp-vpna neighbor 10.0.0.10
```

Configuring Specific Route Export Between Routing Tables

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure the device:

1. Configure the interfaces.

```
[edit interfaces fe-1/3/1]
user@host# set vlan-tagging
user@host# set unit 0 vlan-id 512
user@host# set unit 0 family inet address 10.168.100.3/24
[edit interfaces lo0 unit 0]
user@host# set family inet address 192.168.3.3/32
```

2. Configure the routing policy that specifies particular routes for import into vpna.inet.0 and export from vpna.inet.0.

```
[edit policy-options policy-statement vpna-export]
user@host# set term a from protocol bgp
user@host# set term a then community add vpna-comm
user@host# set term a then accept
user@host# set term b from protocol access-internal
user@host# set term b then accept
user@host# set term c then reject
[edit policy-options policy-statement vpna-import]
user@host# set term a from protocol bgp
user@host# set term a from community vpna-comm
user@host# set term a then accept
user@host# set term b from instance vpna
user@host# set term b from protocol access-internal
```

```
user@host# set term b then accept
user@host# set term c then reject
[edit policy-options]
user@host# set community vpna-comm members target:63000:100
```

3. Configure the routing instance.

```
[edit routing-instances vpna]
user@host# set instance-type vrf
user@host# set interface fe-1/3/1.1
user@host# set route-distinguisher 100:1
user@host# set vrf-import vpna-import
user@host# set vrf-export vpna-export
```

   The **vrf-import** and `vrf-export` statements are used to apply the **vpna-import** and **vpna-export** routing policies.

4. Configure the RIB group, and import routes into the **vpna.inet.2** routing table.

```
[edit routing-options]
user@host# set rib-groups rib-group-vpna-access-internal import-rib vpna.inet.2
```

5. Configure the `auto-export` statement to enable the routes to be exported from one routing table into another.

```
[edit routing-options]
user@host# set auto-export family inet unicast rib-group rib-group-vpna-access-internal
```

6. Configure BGP.

```
[edit routing-instances vpna protocols bgp group bgp-vpna]
user@host# set type external
user@host# set family inet multicast
user@host# set peer-as 100
user@host# set neighbor 100.0.0.10
```

7. Configure the autonomous system (AS) number.

```
[edit routing-options]
user@host# set autonomous-system 63000
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show policy-options`, `show routing-options`, and `show routing-instances` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show interfaces
fe-1/3/1 {
    vlan-tagging;
    unit 0 {
        vlan-id 512;
        family inet {
            address 10.168.100.3/24;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.3.3/32;
        }
    }
}
```

```
user@host# show policy-options
policy-statement vpna-export {
    term a {
        from {
            protocol bgp;
        }
        then {
            community add vpna-comm;
            accept;
        }
    }
    term b {
        from protocol access-internal;
        then accept;
    }
    term c {
        then reject;
    }
}
policy-statement vpna-import {
    term a {
        from {
            protocol bgp;
            community vpna-comm;
        }
        then accept;
    }
    term b {
        from {
            instance vpna;
            protocol access-internal;
        }
        then accept;
    }
    term c {
        then reject;
    }
}
community vpna-comm members target:63000:100;
```

```
user@host# show routing-options
rib-groups {
    rib-group-vpna-access-internal {
        import-rib vpna.inet.2;
    }
}
autonomous-system 63000;
```

```
user@host# show routing-instances
vpna {
    instance-type vrf;
    interface fe-1/3/1.1;
    route-distinguisher 100:1;
    vrf-import vpna-import;
    vrf-export vpna-export;
    routing-options {
        auto-export {
            family inet {
                unicast {
                    rib-group rib-group-vpna-access-internal;
                }
            }
        }
    }
    protocols {
        bgp {
            group bgp-vpna {
                type external;
                family inet {
                    multicast;
                }
                peer-as 100;
                neighbor 100.0.0.10;
            }
        }
    }
}
```

If you are done configuring the device, enter **commit** from configuration mode.

Verification

Confirm that the configuration is working properly by running the `show table route vpna.inet.0` and `show route table vpna.inet.2` commands.

3

CHAPTER

Static Routes

IN THIS CHAPTER

Configure Static Routes

IN THIS SECTION

Understand Basic Static Routing

Static routing is often used when the complexity of a dynamic routing protocol is not desired. A route that does not frequently change, and for which there is only one (or very few) paths to the destination, is a good candidate for static routing. The classic use case for static routing is a single-homed customer attaching to an upstream provider. This type of attachments creates a stub network.

Static routes are defined manually. The route consist of a destination prefix and a next-hop forwarding address. The static route is activated in the routing table and inserted into the forwarding table when the next-hop address is reachable. Traffic that matches the static route is forwarded to the specified next-hop address.

You can specify options that define additional information about static routes. These attributes, for example a community tag or a route metric, are included with the route when it's installed in the routing table. These additional route attributes are not required for basic static routing.

NOTE: You can add only well-known and normal communities to a static route that falls in the range 65535:65535. You can only specify a community value in a static route, not a community name. We do not support adding extended communities to a static route. For more information, see Understanding BGP Communities, Extended Communities, and Large Communities as Routing Policy Match Conditions and Understanding How to Define BGP Communities and Extended Communities.

Example: Configure IPv4 Static Routing for a Stub Network

IN THIS SECTION

NOTE: Our content testing team has validated and updated this example.

This example shows how to configure basic static routing for IPv4.

Requirements

Two devices running Junos OS with a shared network link. No special configuration beyond basic device initialization (management interface, remote access, user login accounts, and so on), is required before you configure this example.

IPv4 Static Routing Overview

IN THIS SECTION

There are many practical applications for static routes. Static routing is often used at the network edge to support attachment to stub networks. Stub networks have a single point of entry and egress, making them well suited to the simplicity of a static route. In Junos OS, static routes have a global preference (administrative distance) of 5. This value makes them preferred over routes learned from dynamic protocols like OSPF or BGP.

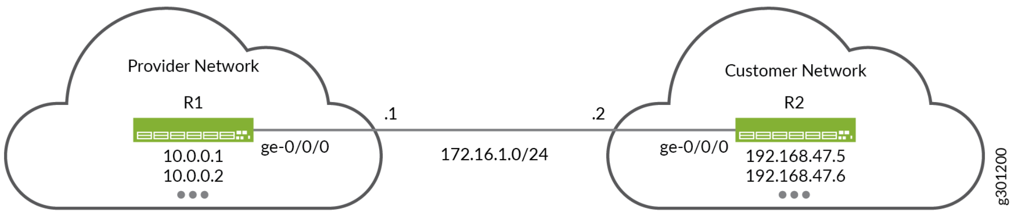
IPv4 Static Routing Topology

Figure 1 on page 24 shows the example topology.

In this example, you configure the static route 192.168.47.0/24 on the provider device (R1), using a next-hop address of 172.16.1.2. This route allows the provider device to reach the remote networks at the customer site. You also configure a static default route of 0.0.0.0/0 on the customer device (R2), using a next-hop address of 172.16.1.1. The default route ensures the customer can reach all nonlocal networks by forwarding this traffic to the provider network.

Multiple loopback addresses are configured on both devices. These loopback addresses provide remote destinations to ping, so you can verify the IPv4 static routing works properly.

Figure 1: IPv4 Stub Network Connected to a Service Provider



IPv4 Static Route Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure basic IPv4 static routing on the R1 and R2 devices, edit the following commands as needed and paste them into the CLI at the `[edit]` hierarchy level. Be sure to issue a `commit` from configuration mode to activate the changes.

R1 Device (Provider)

```
set system host-name R1
set interfaces ge-0/0/0 unit 0 description “Link from R1 to R2”
set interfaces ge-0/0/0 unit 0 family inet address 172.16.1.1/24
set interfaces lo0 unit 0 family inet address 10.0.0.1/32
set interfaces lo0 unit 0 family inet address 10.0.0.2/32
set routing-options static route 192.168.47.0/24 next-hop 172.16.1.2
```

R2 Device (Customer)

```
set system host-name R2
set interfaces ge-0/0/0 unit 0 description “Link from R2 to R1”
set interfaces ge-0/0/0 unit 0 family inet address 172.16.1.2/24
set interfaces lo0 unit 0 family inet address 192.168.47.5/32
set interfaces lo0 unit 0 family inet address 192.168.47.6/32
set routing-options static route 0.0.0.0/0 next-hop 172.16.1.1
```

Configure the R1 and R2 Devices

Step-by-Step Procedure

This example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure basic static routes:

1. Configure the hostname on the R1 (provider) device.

   ```
   [edit ]
   user@R1# set system host-name R1
   ```

2. Configure the interfaces on the R1 (provider) device.

   ```
   [edit interfaces]
   user@R1# set ge-0/0/0 unit 0 description "Link from R1 to R2"
   user@R1# set ge-0/0/0 unit 0 family inet address 172.16.1.1/24
   user@R1# set lo0 unit 0 family inet address 10.0.0.1/32
   user@R1# set lo0 unit 0 family inet address 10.0.0.2/32
   ```

3. Define the static route to the customer’s prefix on the R1 device. Be sure to specify the R2 end of the point-to-point link as the next hop for the static route.

The static route ensures the provider network can route to all remote destinations in the customer network by forwarding traffic through the R2 device.

```
[edit routing-options]
user@R1# set static route 192.168.47.0/24 next-hop 172.16.1.2
```

4. Commit your changes on the R1 device.

```
[edit ]
user@R1# commit
```

5. Configure the hostname on the R2 (customer) device.

```
[edit ]
user@R2# set system host-name R2
```

6. Configure the interfaces on the R2 (customer) device.

```
[edit interfaces]
user@R2# set ge-0/0/0 unit 0 description "Link from R2 to R1"
user@R2# set ge-0/0/0 unit 0 family inet address 172.16.1.2/24
user@R2# set lo0 unit 0 family inet address 192.168.47.5/32
user@R2# set lo0 unit 0 family inet address 192.168.47.6/32
```

7. Define the IPv4 static default route on the R2 device. Be sure to specify the R1 end of the point-to-point link as the next hop for the static route.

   The IPv4 default route ensures the customer can route to all nonlocal destinations by forwarding traffic to the R1 device in the provider network.

```
[edit routing-options]
user@R2# set static route 0.0.0.0/0 next-hop 172.16.1.1
```

8. Commit your changes on the R2 device.

```
[edit]
user@R2# commit
```

Results

Confirm your configuration by issuing the `show interfaces` and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

R1 Device

```
user@R1# show interfaces
ge-0/0/0 {
    unit 0 {
        description "Link from R1 to R2";
        family inet {
            address 172.16.1.1/24;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 10.0.0.1/32;
            address 10.0.0.2/32;
        }
    }
}
```

```
user@R1# show routing-options
static {
    route 192.168.47.0/24 next-hop 172.16.1.2;
}
```

R2 Device

```
user@R2# show interfaces
ge-0/0/0 {
    unit 0 {
        description "Link from R2 to R1";
        family inet {
            address 172.16.1.2/24;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.47.5/32;
            address 192.168.47.6/32;
        }
    }
}
```

```
user@R2# show routing-options
static {
    route 0.0.0.0/0 next-hop 172.16.1.1;
}
```

Verification

IN THIS SECTION

Confirm your IPv4 static routing is working properly.

Check the Routing Tables

Purpose

Confirm the IPv4 static routes are listed as active in the routing tables of both devices.

Action

```
user@R1> show route
inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
10.0.0.1/32        *[Direct/0] 00:29:43
                    > via lo0.0
10.0.0.2/32        *[Direct/0] 00:29:43
                    > via lo0.0
172.16.1.0/24      *[Direct/0] 00:34:40
                    > via ge-0/0/0.0
172.16.1.1/32      *[Local/0] 00:34:40
                      Local via ge-0/0/0.0
192.168.47.0/24    *[Static/5] 00:31:23
                    > to 172.16.1.2 via ge-0/0/0.0
```

```
user@R2> show route
inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 00:31:24
                    > to 172.16.1.1 via ge-1/2/0.1
172.16.1.0/24      *[Direct/0] 00:35:21
                    > via ge-0/0/0.0
172.16.1.2/32      *[Local/0] 00:35:21
                      Local via ge-0/0/0.0
192.168.47.5/32    *[Direct/0] 00:35:22
                    > via lo0.0
192.168.47.6/32    *[Direct/0] 00:35:21
                    > via lo0.0
```

Meaning

The output confirms the static routes are present in the routing tables of both devices. The `*` symbol indicates the routes are active. The next hop for the static routes correctly point to the IP address assigned to the remote end of the link.

Ping the Remote Loopback Addresses

Purpose

Verify that the IPv4 static routes provide connectivity between the loopback addresses of both devices. It's a good idea to source your test traffic from a loopback address on the local device using the `source` option. This approach validates forwarding between the loopback addresses of both devices in a single command.

From the R1 device, ping a loopback interface address on the R2 device.

From the R2 device, ping a loopback interface address on the R1 device.

Action

```
user@R1> ping 192.168.47.5 count 2 source 10.0.0.1
PING 192.168.47.5 (192.168.47.5): 56 data bytes
64 bytes from 192.168.47.5: icmp_seq=0 ttl=64 time=1.344 ms
64 bytes from 192.168.47.5: icmp_seq=1 ttl=64 time=1.279 ms

--- 192.168.47.5 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.279/1.312/1.344/0.032 ms
```

```
user@R2> ping 10.0.0.1 count 2 source 192.168.47.5
PING 10.0.0.1 (10.0.0.1): 56 data bytes
64 bytes from 10.0.0.1: icmp_seq=0 ttl=64 time=1.939 ms
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=2.139 ms

--- 10.0.0.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.939/2.039/2.139/0.100 ms
```

Meaning

The output confirms the static routes allow traffic to be forwarded between the provider and customer networks.

Example: Configure IPv6 Static Routing for a Stub Network

IN THIS SECTION

NOTE: Our content testing team has validated and updated this example.

This example shows how to configure basic static routes for IPv6.

Requirements

Two devices running Junos OS with a shared network link. No special configuration beyond basic device initialization (management interface, remote access, user login accounts, and so on), is required before you configure this example.

IPv6 Static Routing Overview

IN THIS SECTION

There are many practical applications for static routes. Static routing is often used at the network edge to support attachment to stub networks, which, given their single point of entry and egress, are well suited to the simplicity of a static route. In Junos OS, static routes have a global preference of 5. Static routes are activated when the specified next hop is reachable.

You can specify options that define additional information about static IPv6 routes. These attributes, for example a community tag or route metric, are included with the route when it's installed in the routing table. These additional route attributes are not required for basic IPv6 static routing.

IPv6 Static Routing Topology

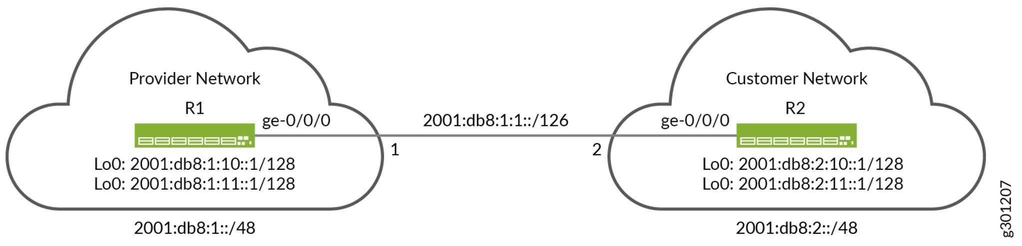
Figure 2 on page 32 provides the IPv6 static routing topology.

In this example the provider and customer networks have been allocated the IPv6 prefixes 2001:db8:1::/48 and 2001:db8:2::/48, respectively. Both networks are free to allocate longer prefixes (subnetworks) from their assigned prefix block. The point-to-point link is numbered from the provider's address space using a /126 prefix length. Each device has two loopback addresses allocated from their assigned prefix using a /128 prefix length.

You configure a static route to the customer prefix (2001:db8:2::/48) on the provider (R1) network device, using a next hop of 2001:db8:1:1::2. This route provides reachability from the provider device to the remote networks at the customer site. On the customer device (R2), you configure a static default route of ::/0, using a next-hop address 2001:db8:1:1::1. The default route provides the customer with reachability to all nonlocal prefixes through the provider's network.

Multiple loopback addresses are configured on both devices. These loopback addresses provide remote destinations to ping, allowing you to verify the IPv6 static routing works properly.

Figure 2: IPv6 Stub Network Connected to a Service Provider



IPv6 Static Route Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure basic IPv6 static routing on the R1 and R2 devices, edit the following commands as needed and paste them into the CLI at the `[edit]` hierarchy level. Be sure to issue a `commit` from configuration mode to activate the changes.

R1 Device (Provider)

```
set system host-name R1
set interfaces ge-0/0/0 description "Link from R1 to R2"
set interfaces ge-0/0/0 unit 0 family inet6 address 2001:db8:1:1::1/126
set interfaces lo0 unit 0 family inet6 address 2001:db8:1:10::1/128
set interfaces lo0 unit 0 family inet6 address 2001:db8:1:11::1/128
set routing-options rib inet6.0 static route 2001:db8:2::/48 next-hop 2001:db8:1:1::2
```

R2 Device (Customer)

```
set system host-name R2
set interfaces ge-0/0/0 description "Link from R2 to R1"
set interfaces ge-0/0/0 unit 0 family inet6 address 2001:db8:1:1::2/126
set interfaces lo0 unit 0 family inet6 address 2001:db8:2:10::1/128
set interfaces lo0 unit 0 family inet6 address 2001:db8:2:11::1/128
set routing-options rib inet6.0 static route ::/0 next-hop 2001:db8:1:1::1
```

Configure the R1 and R2 Devices

Step-by-Step Procedure

This example that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

Follow these steps to configure basic IPv6 static routes:

1. Configure the hostname on the R1 (provider) device.

   ```
   [edit ]
   user@R1# set system host-name R1
   ```

2. Configure the interfaces on the R1 (provider) device.

```
[edit interfaces]
user@R1# set ge-0/0/0 description "Link from R1 to R2"
user@R1# set ge-0/0/0 unit 0 family inet6 address 2001:db8:1:1::1/126
user@R1# set lo0 unit 0 family inet6 address 2001:db8:1:10::1/128
user@R1# set lo0 unit 0 family inet6 address 2001:db8:1:11::1/128
```

3. Define the static route to the customer's IPv6 prefix on the R1 device. Be sure to set the next-hop address to the customer end of the point-to-point link.

   The use of a /48 bit prefix length ensures that the R1 device can reach all possible remote destinations in the customer network by forwarding through the R2 device.

```
[edit routing-options]
user@R1# set rib inet6.0 static route 2001:db8:2::/48 next-hop 2001:db8:1:1::2
```

4. Commit your changes on the R1 device.

```
[edit ]
user@R1# commit
```

5. Configure the hostname on the R2 (customer) device.

```
[edit ]
user@R2# set system host-name R2
```

6. Configure the interfaces on the R2 (customer) device.

```
[edit interfaces]
user@R2# set ge-0/0/0 description "Link from R2 to R1"
user@R2# set ge-0/0/0 unit 0 family inet6 address 2001:db8:1:1::2/126
user@R2# set lo0 unit 0 family inet6 address 2001:db8:2:10::1/128
user@R2# set lo0 unit 0 family inet6 address 2001:db8:2:10::2/128
```

7. Define the IPv6 static default route on the R2 device. Be sure to set the next-hop address to the provider end of the point-to-point link.

The IPv6 default route ensures that the R2 device can reach all nonlocal destinations by forwarding traffic through the R1 device in the provider network.

```
[edit routing-options]
user@R2# set rib inet6.0 static route ::/0 next-hop 2001:db8:1:1::1
```

8. Commit your changes on the R2 device.

```
[edit]
user@R2# commit
```

Results

Confirm your configuration by issuing the `show interfaces` and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

R1 Device

```
user@R1# show interfaces
ge-0/0/0 {
    description "Link from R1 to R2";
    unit 0 {
        family inet6 {
            address 2001:db8:1:1::1/126;
        }
    }
}
lo0 {
    unit 0 {
        family inet6 {
            address 2001:db8:1:10::1/128;
            address 2001:db8:1:11::1/128;
        }
    }
}
```

```
user@R1# show routing-options
rib inet6.0 {
    static {
        route 2001:db8:2::/48 next-hop 2001:db8:1:1::2;
    }
}
```

R2 Device

```
user@R2# show interfaces
ge-0/0/0 {
    description "Link from R2 to R1";
    unit 0 {
        family inet6 {
            address 2001:db8:1:1::2/126;
        }
    }
}
lo0 {
    unit 0 {
        family inet6 {
            address 2001:db8:2:10::1/128;
            address 2001:db8:2:11::1/128;
        }
    }
}
```

```
user@R2# show routing-options
rib inet6.0 {
    static {
        route ::/0 next-hop 2001:db8:1:1::1;
    }
}
```

Verification

IN THIS SECTION

Confirm IPv6 static routing works properly.

Checking the Routing Tables

Purpose

Verify the IPv6 static routes are active in the routing tables of both devices.

Action

```
user@R1> show route protocol static
inet6.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2001:db8:2::/48    *[Static/5] 02:07:11
                    >  to 2001:db8:1:1::2 via ge-0/0/0.0
```

```
user@R2> show route protocol static
inet6.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

::/0               *[Static/5] 02:13:56
                    >  to 2001:db8:1:1::1 via ge-0/0/0.0
```

Meaning

The output confirms the IPv6 static routes are present in the routing tables of both devices. The `*` symbol indicates the routes are active. Both the static routes correctly point to the remote end of the point-to-point link as the next hop for matching traffic.

Ping the Remote Loopback Addresses

Purpose

Verify that the IPv6 static routes provide connectivity between the loopback addresses of both devices. It's a good idea to source your test traffic from a loopback address on the local device using the `source` option. This approach validates forwarding between the loopback addresses of both devices in a single command.

From the R1 device, ping a loopback address on the R2 device.

From the R2 device, ping q loopback address on the R1 device.

Action

```
user@R1> ping 2001:db8:2:10::1 source 2001:db8:1:10::1 count 2
PING6(56=40+8+8 bytes) 2001:db8:1:10::1 --> 2001:db8:2:10::1
16 bytes from 2001:db8:2:10::1, icmp_seq=0 hlim=64 time=2.770 ms
16 bytes from 2001:db8:2:10::1, icmp_seq=1 hlim=64 time=2.373 ms

--- 2001:db8:2:10::1 ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 2.373/2.572/2.770/0.198 ms
```

```
user@R2> ping 2001:db8:1:10::1 source 2001:db8:2:10::1 count 2
PING6(56=40+8+8 bytes) 2001:db8:2:10::1 --> 2001:db8:1:10::1
16 bytes from 2001:db8:1:10::1, icmp_seq=0 hlim=64 time=1.985 ms
16 bytes from 2001:db8:1:10::1, icmp_seq=1 hlim=64 time=1.704 ms

--- 2001:db8:1:10::1 ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 1.704/1.845/1.985/0.140 ms
```

Meaning

The output confirms the IPv6 static routes allow traffic to be forwarded between the provider and customer networks.

Static Route Preferences and Qualified Next Hops

IN THIS SECTION

Understanding Static Route Preferences and Qualified Next Hops

A static route destination address can have multiple next hops associated with it. In this case, multiple routes are inserted into the routing table, and route selection must occur. Because the primary criterion for route selection is the *route preference*, you can control the routes that are used as the primary route for a particular destination by setting the route preference associated with a particular next hop. The routes with a lower route preference are always used to route traffic. When you do not set a preferred route, the Junos OS chooses in a random fashion one of the next-hop addresses to install into the forwarding table.

In general, the default properties assigned to a static route apply to all the next-hop addresses configured for the static route. If, however, you want to configure two possible next-hop addresses for a particular route and have them treated differently, you can define one as a qualified next hop.

Qualified next hops allow you to associate one or more properties with a particular next-hop address. You can set an overall preference for a particular static route and then specify a different preference for the qualified next hop. For example, suppose two next-hop addresses (10.10.10.10 and 10.10.10.7) are associated with the static route 192.168.47.5/32. A general preference is assigned to the entire static route, and then a different preference is assigned to only the qualified next-hop address 10.10.10.7. For example:

```
route 192.168.47.5/32 {
    next-hop 10.10.10.10;
    qualified-next-hop 10.10.10.7 {
        preference 6;
    }
    preference 5;
}
```

In this example, the qualified next hop 10.10.10.7 is assigned the preference 6, and the next-hop 10.10.10.10 is assigned the preference 5.

NOTE: The `preference` and `metric` options in the `[edit route` *`route`* `qualified-next-hop]` hierarchy only apply to the qualified next hops. The qualified next-hop preference and metric override the route preference and metric for that specific qualified next hop only, similar to how the route preference overrides the default preference and metric (for that specific route).

NOTE: Starting in Junos OS Release 15.1R4, the router no longer supports a configuration where a static route points to a next hop that is tied to a subscriber. Typically, this might occur when RADIUS assigns the next hop with the Framed-IP-Address attribute. An alternative to this misconfiguration is to have the RADIUS server provide a Framed-Route attribute that matches the static route.

Example: Configuring Static Route Preferences and Qualified Next Hops to Control Static Route Selection

IN THIS SECTION

This example shows how to control static route selection.

Requirements

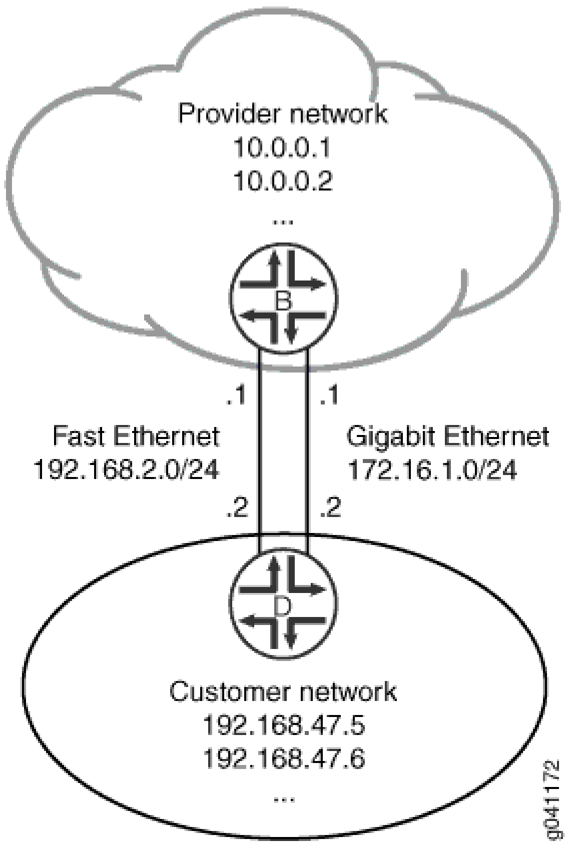
In this example, no special configuration beyond device initialization is required.

Overview

IN THIS SECTION

In this example, the static route 192.168.47.0/24 has two possible next hops. Because one link has higher bandwidth, this link is the preferred path. To enforce this preference, the `qualified-next-hop` statement is included in the configuration on both devices. See Figure 3 on page 41.

Figure 3: Controlling Static Route Selection



Topology

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

Device B in Provider Network

```
set interfaces ge-1/2/0 unit 0 description B->D
set interfaces ge-1/2/0 unit 0 family inet address 172.16.1.1/24
set interfaces fe-1/2/1 unit 2 description secondary-B->D
set interfaces fe-1/2/1 unit 2 family inet address 192.168.2.1/24
set interfaces lo0 unit 57 family inet address 10.0.0.1/32
set interfaces lo0 unit 57 family inet address 10.0.0.2/32
set routing-options static route 192.168.47.0/24 next-hop 172.16.1.2
set routing-options static route 192.168.47.0/24 qualified-next-hop 192.168.2.2 preference 25
```

Device D in Customer Network

```
set interfaces ge-1/2/0 unit 1 description D->B
set interfaces ge-1/2/0 unit 1 family inet address 172.16.1.2/24
set interfaces fe-1/2/1 unit 3 description secondary-D->B
set interfaces fe-1/2/1 unit 3 family inet address 192.168.2.2/24
set interfaces lo0 unit 2 family inet address 192.168.47.5/32
set interfaces lo0 unit 2 family inet address 192.168.47.6/32
set routing-options static route 0.0.0.0/0 next-hop 172.16.1.1
set routing-options static route 0.0.0.0/0 qualified-next-hop 192.168.2.1 preference 25
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To control static route selection:

1. On Device B, configure the interfaces.

```
[edit interfaces]
user@B# set ge-1/2/0 unit 0 description B->D
user@B# set ge-1/2/0 unit 0 family inet address 172.16.1.1/24
user@B# set fe-1/2/1 unit 2 description secondary-B->D
user@B# set fe-1/2/1 unit 2 family inet address 192.168.2.1/24
user@B# set lo0 unit 57 family inet address 10.0.0.1/32
user@B# set lo0 unit 57 family inet address 10.0.0.2/32
```

2. On Device B, configure a static route to the customer network.

```
[edit routing-options static route 192.168.47.0/24]
user@B# set next-hop 172.16.1.2
```

3. On Device B, configure a backup route to the customer network.

```
[edit routing options static route 192.168.47.0/24]
user@B# set qualified-next-hop 192.168.2.2 preference 25
```

4. On Device D, configure the interfaces.

```
[edit interfaces]
user@D# set ge-1/2/0 unit 1 description D->B
user@D# set ge-1/2/0 unit 1 family inet address 172.16.1.2/24
user@D# set fe-1/2/1 unit 3 description secondary-D->B
user@D# set fe-1/2/1 unit 3 family inet address 192.168.2.2/24
user@D# set lo0 unit 2 family inet address 192.168.47.5/32
user@D# set lo0 unit 2 family inet address 192.168.47.6/32
```

5. On Device D, configure a static default route to external networks.

```
[edit routing options static route 0.0.0.0/0]
user@D# set next-hop 172.16.1.1
```

6. On Device D, configure a backup static default route to external networks.

```
[edit routing options static route 0.0.0.0/0]
user@D# set qualified-next-hop 192.168.2.1 preference 25
```

Results

Confirm your configuration by issuing the `show interfaces` and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@B# show interfaces
ge-1/2/0 {
    unit 0 {
        description B->D;
        family inet {
            address 172.16.1.1/24;
        }
    }
}
fe-1/2/1 {
    unit 2 {
        description secondary-B->D;
        family inet {
            address 192.168.2.1/24;
        }
    }
}
lo0 {
    unit 57 {
        family inet {
            address 10.0.0.1/32;
            address 10.0.0.2/32;
        }
    }
}
```

```
user@B# show routing-options
static {
    route 192.168.47.0/24 {
        next-hop 172.16.1.2;
        qualified-next-hop 192.168.2.2 {
            preference 25;
        }
    }
}
```

```
user@D# show interfaces
ge-1/2/0 {
    unit 1 {
        description D->B;
        family inet {
            address 172.16.1.2/24;
        }
    }
}
fe-1/2/1 {
    unit 3 {
        description secondary-D->B;
        family inet {
            address 192.168.2.2/24;
        }
    }
}
lo0 {
    unit 2 {
        family inet {
            address 192.168.47.5/32;
            address 192.168.47.6/32;
        }
    }
}
```

```
user@D# show routing-options
static {
    route 0.0.0.0/0 {
        next-hop 172.16.1.1;
        qualified-next-hop 192.168.2.1 {
            preference 25;
        }
    }
}
```

If you are done configuring the devices, enter **commit** from configuration mode on both devices.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Checking the Routing Tables

Purpose

Make sure that the static routes appear in the routing tables of Device B and Device D.

Action

```
user@B> show route protocol static
inet.0: 7 destinations, 8 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.47.0/24    *[Static/5] 02:02:03
                    > to 172.16.1.2 via ge-1/2/0.0
                    [Static/25] 01:58:21
                    > to 192.168.2.2 via fe-1/2/1.2
```

```
user@D> show route protocol static
inet.0: 7 destinations, 8 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 02:02:12
                    > to 172.16.1.1 via ge-1/2/0.1
                    [Static/25] 01:58:31
                    > to 192.168.2.1 via fe-1/2/1.3
```

Meaning

The asterisks (*) in the routing tables show the active routes. The backup routes are listed next.

Pinging the Remote Addresses

Purpose

Verify that the static routes are working.

From Device B, ping one of the loopback interface addresses on Device D.

From Device D, ping one of the loopback interface addresses on Device B.

Action

```
user@B> ping 192.168.47.5
PING 192.168.47.5 (192.168.47.5): 56 data bytes
64 bytes from 192.168.47.5: icmp_seq=0 ttl=64 time=156.126 ms
```

```
64 bytes from 192.168.47.5: icmp_seq=1 ttl=64 time=120.393 ms
64 bytes from 192.168.47.5: icmp_seq=2 ttl=64 time=175.361 ms
```

```
user@D> ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1): 56 data bytes
64 bytes from 10.0.0.1: icmp_seq=0 ttl=64 time=1.315 ms
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=31.819 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=1.268 ms
```

Making Sure That the Backup Route Becomes the Active Route

Purpose

If the primary route becomes unusable, make sure that the backup secondary route becomes active.

Action

1. Disable the active route by deactivating the ge-1/2/0.0 interface on Device B.

   ```
   user@B# deactivate interfaces ge-1/2/0 unit 0 family inet address 172.16.1.1/24
   user@B# commit
   ```

2. Check Device B's routing table.

   ```
   user@B> show route protocol static
   inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
   + = Active Route, - = Last Active, * = Both

   192.168.47.0/24    *[Static/25] 02:06:24
                       > to 192.168.2.2 via fe-1/2/1.2
   ```

Meaning

The backup route has become the active route.

Conserving IP Addresses Using Static Routes

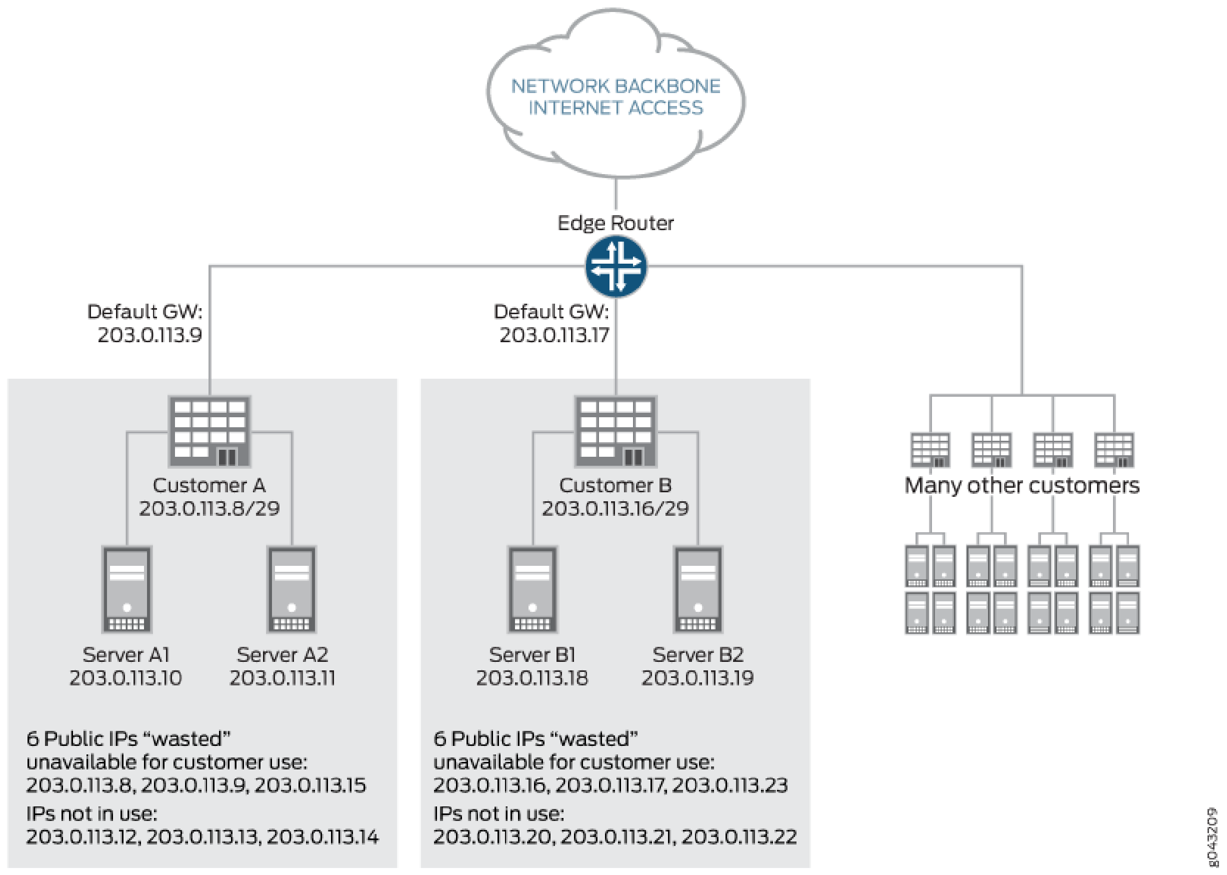
IN THIS SECTION

Hosting providers host multiple servers for multiple customers and want to conserve the usage of their IP address space. Traditionally, when a hosting provider client adds new servers, the servers are allocated a small block of IP addresses, such as a /29 block, and the client's servers are all located in that block of IP addresses.

The Issue, Illustrated

For example, Customer A might need three servers and is assigned the block 10.3.3.0/29 (10.3.3.0 through 10.3.3.7). In this scenario, several IP addresses are consumed. These include the network and broadcast IP addresses (10.3.3.0 and 10.3.3.7), the addresses for the router gateway that the servers are connected to, and the addresses of the individual servers. To allocate three servers, eight IP addresses have to be allocated. Breaking up a single /24 network into 32 /29 networks results in 96 IP addresses out of the 256, in that /24 is being consumed by the network, broadcast, and gateway addresses. When this effect is multiplied across thousands of hosting providers, IP address space is far from being used efficiently. Figure 4 on page 50 illustrates the issue.

Figure 4: Inefficient Use of IP Address Space


NETWORK BACKBONE
INTERNET ACCESS
Edge Router
Default GW: 203.0.113.9
Default GW: 203.0.113.17
Customer A 203.0.113.8/29
Customer B 203.0.113.16/29
Many other customers
Server A1 203.0.113.10
Server A2 203.0.113.11
Server B1 203.0.113.18
Server B2 203.0.113.19
6 Public IPs "wasted" unavailable for customer use: 203.0.113.8, 203.0.113.9, 203.0.113.15
IPs not in use: 203.0.113.12, 203.0.113.13, 203.0.113.14
6 Public IPs "wasted" unavailable for customer use: 203.0.113.16, 203.0.113.17, 203.0.113.23
IPs not in use: 203.0.113.20, 203.0.113.21, 203.0.113.22
g043209

In this configuration, each customer is allocated a /29 block of address space. For each block, the network, broadcast, and gateway addresses are not available for server IP addressing, which results in three IP addresses being used inefficiently. In addition, the blocks consume unused IP addresses for future expansion.

Solution

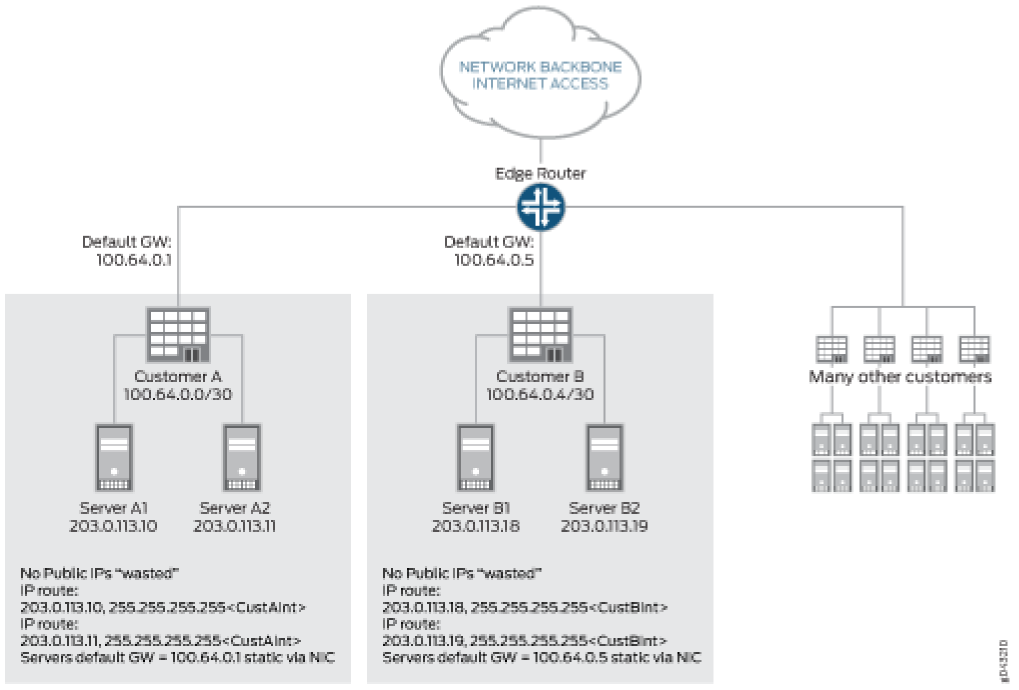
This issue can be resolved by configuring the interface on the router with an address from the reserved IPv4 prefix for shared address space (RFC 6598) and by using static routes pointed at interfaces. IANA has recorded the allocation of an IPv4 /10 for use as shared address space. The shared address space address range is 100.64.0.0/10.

The interface in the router gets allocated an IP address from the RFC 6598 space, so it is not consuming publicly routable address space, and connectivity is handled with static routes on an interface. The interface in the server is configured with a publicly routable address, but the router interfaces are not. Network and broadcast addresses are consumed out of the RFC 6598 space rather than the publicly routable address space.

This feature is supported on QFX10000 switches starting with Junos OS 17.1R1.

Figure 5 on page 51 shows the efficient use of IP address space.

Figure 5: Configuration Using the Shared Address Space



In this configuration, each customer gets allocated individual IP addresses per server. There is a static route that can be configured as a host route. The interface in the router gets allocated an IP address from the RFC 6598 space, so it does not consume publicly routable address space, and connectivity is handled with static routes out to an interface.

Configuration

The configuration would look like this for Customer A on the gateway router:

```
interfaces {
    ge-1/0/1 {
        unit 0 {
            family inet {
                address 100.64.0.1/30;
            }
        }
        }
    }
```

```
    routing-options {
        static {
            route 203.0.113.10/32 {
                qualified-next-hop ge-1/0/1.0;
            }
            route 203.0.113.11 {
                qualified-next-hop ge-1/0/1.0;
            }
        }
    }
```

With this configuration, no publicly routable IP addresses are wasted. It is worth noting that when a packet is forwarded in this configuration from the router to the server of Customer A's server 203.0.113.10, the route is forwarded out to the interface ge-1/0/1.0 which has an IP address of `100.64.0.1`.

The servers for customer A would be configured as follows:

```
ifconfig eth0 203.0.113.10 netmask 255.255.255.255
route add -host 100.64.0.1/32 dev eth0 route add default gw 100.64.0.1
```

```
ifconfig eth0 203.0.113.11 netmask 255.255.255.255
route add -host 100.64.0.1/32 dev eth0 route add default gw 100.64.0.1
```

This example shows a single host route per server, which is a 1:1 mapping. This could equate to a large number of static host routes, if maintained. For scaling purposes, we need to support nonhost routes in this environment. For example, if there were a Customer C in this configuration that had eight servers, it would be much more efficient to allocate a /29 route on the router that points out the interface on which the eight servers are connected. If Customer C were allocated server IPs from 203.0.114.8 through 203.0.114.15 and these were connected via interface ge-1/0/2.0, this would look like:

```
user@host# set routing-options static route 203.0.114.8/29 qualified-next-hop ge-1/0/2.0
```

Understanding Static Route Control in Routing and Forwarding Tables

IN THIS SECTION

You can control the importation of static routes into the routing and forwarding tables in a number of ways. Primary ways include assigning one or more of the following attributes to the route:

- **retain**—Keeps the route in the forwarding table after the routing process shuts down or the device reboots.
- **no-readvertise**—Prevents the route from being readvertised to other routing protocols.
- **passive**—Rejects traffic destined for the route.

This topic includes the following sections:

Route Retention

By default, static routes are not retained in the forwarding table when the routing process shuts down. When the routing process starts up again, any routes configured as static routes must be added to the forwarding table again. To avoid this latency, routes can be flagged as **retain**, so that they are kept in the forwarding table even after the routing process shuts down. Retention ensures that the routes are always in the forwarding table, even immediately after a system reboot.

Readvertisement Prevention

Static routes are eligible for readvertisement by other routing protocols by default. In a stub area where you might not want to readvertise these static routes under any circumstances, you can flag the static routes as **no-readvertise**.

Forced Rejection of Passive Route Traffic

Generally, only active routes are included in the routing and forwarding tables. If a static route's next-hop address is unreachable, the route is marked **passive**, and it is not included in the routing or forwarding tables. To force a route to be included in the routing tables regardless of next-hop reachability, you can flag the route as **passive**. If a route is flagged **passive** and its next-hop address is unreachable, the route is included in the routing table, and all traffic destined for the route is rejected.

Example: Preventing a Static Route from Being Readvertised

IN THIS SECTION

This example shows how to prevent a static route from being readvertised into OSPF, thereby preventing the route from appearing in the routing and forwarding tables.

Requirements

In this example, no special configuration beyond device initialization is required.

Overview

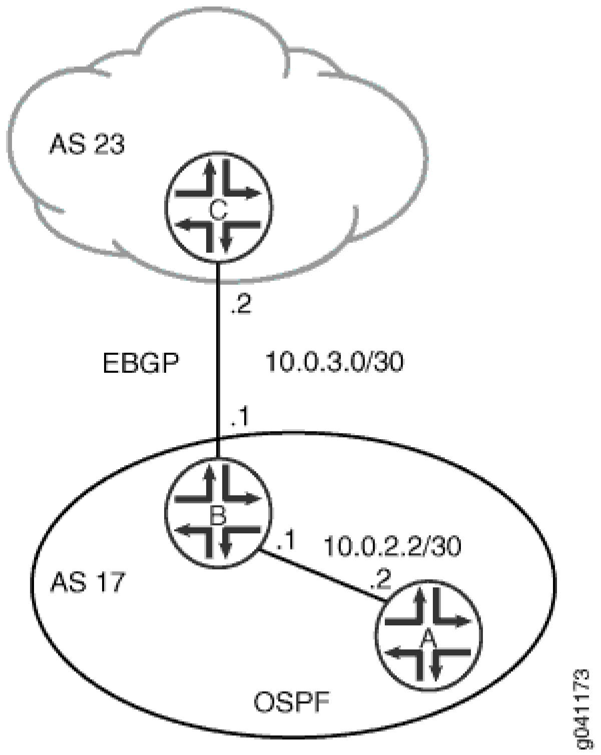
IN THIS SECTION

This example shows how to configure a routing policy that readvertises static routes into OSPF, with the exception of one static route that is not readvertised because it is tagged with the `no-readvertise` statement.

Topology

Figure 6 on page 55 shows the sample network.

Figure 6: Customer Routes Connected to a Service Provider



Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

Device A

```
set interfaces fe-1/2/0 unit 4 description A->B
set interfaces fe-1/2/0 unit 4 family inet address 10.0.2.2/30
set protocols ospf area 0.0.0.0 interface fe-1/2/0.4
```

Device B

```
set interfaces fe-1/2/0 unit 3 description B->A
set interfaces fe-1/2/0 unit 3 family inet address 10.0.2.1/30
set interfaces fe-1/2/1 unit 6 description B->C
set interfaces fe-1/2/1 unit 6 family inet address 10.0.3.1/30
set protocols bgp group ext type external
set protocols bgp group ext peer-as 23
set protocols bgp group ext neighbor 10.0.3.2
set protocols ospf export send-static
set protocols ospf area 0.0.0.0 interface fe-1/2/0.3
set policy-options policy-statement send-static from protocol static
set policy-options policy-statement send-static then accept
set routing-options static route 0.0.0.0/0 next-hop 10.0.3.2
set routing-options static route 192.168.0.0/24 next-hop 10.0.3.2
set routing-options static route 192.168.0.0/24 no-readvertise
set routing-options autonomous-system 17
```

Device C

```
set interfaces fe-1/2/0 unit 7 description B->C
set interfaces fe-1/2/0 unit 7 family inet address 10.0.3.2/30
set interfaces lo0 unit 5 family inet address 192.168.0.1/32
set protocols bgp group ext type external
set protocols bgp group ext peer-as 17
set protocols bgp group ext neighbor 10.0.3.1
set routing-options autonomous-system 23
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure Device A:

1. Configure the interface to Device B.

```
[edit interfaces fe-1/2/0 unit 4]
user@A# set description A->B
user@A# set family inet address 10.0.2.2/30
```

2. Configure OSPF to form an OSPF peer relationship with Device B.

```
[edit protocols ospf area 0.0.0.0]
user@A# set interface fe-1/2/0.4
```

Step-by-Step Procedure

To configure Device B:

1. Configure the interfaces to Device A and Device C.

```
[edit interfaces]
user@B# set fe-1/2/0 unit 3 description B->A
user@B# set fe-1/2/0 unit 3 family inet address 10.0.2.1/30
user@B# set fe-1/2/1 unit 6 description B->C
user@B# set fe-1/2/1 unit 6 family inet address 10.0.3.1/30
```

2. Configure one or more static routes and the autonomous system (AS) number.

```
[edit routing-options]
user@B# set static route 0.0.0.0/0 next-hop 10.0.3.2
user@B# set static route 192.168.0.0/24 next-hop 10.0.3.2
user@B# set autonomous-system 17
```

3. Configure the routing policy.

This policy exports static routes from the routing table into OSPF.

```
[edit policy-options policy-statement send-static]
user@B# set from protocol static
user@B# set then accept
```

4. Include the `no-readvertise` statement to prevent the 192.168.0.0/24 route from being exported into OSPF.

```
[edit routing-options]
user@B# set static route 192.168.0.0/24 no-readvertise
```

5. Configure the routing protocols.

   The BGP configuration forms an external BGP (EBGP) peer relationship with Device C.

   The OSPF configuration forms an OSPF peer relationship with Device A and applies the **send-static** routing policy.

```
[edit protocols]
user@B# set bgp group ext type external
user@B# set bgp group ext peer-as 23
user@B# set bgp group ext neighbor 10.0.3.2
user@B# set ospf export send-static
user@B# set ospf area 0.0.0.0 interface fe-1/2/0.3
```

Step-by-Step Procedure

To configure Device C:

1. Create the interface to Device B, and configure the loopback interface.

```
[edit interfaces ]
user@C# set fe-1/2/0 unit 7 description B->C
user@C# set fe-1/2/0 unit 7 family inet address 10.0.3.2/30
user@C# set lo0 unit 5 family inet address 192.168.0.1/32
```

2. Configure the EBGP peering session with Device B.

```
[edit protocols bgp group ext]
user@C# set type external
user@C# set peer-as 17
user@C# set neighbor 10.0.3.1
```

3. Configure the AS number.

```
[edit routing-options]
user@C# set autonomous-system 23
```

Results

Confirm your configuration by issuing the `show interfaces`, `show policy-options`, `show protocols`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device A

```
user@A# show interfaces
fe-1/2/0 {
    unit 4 {
        description A->B;
        family inet {
            address 10.0.2.2/30;
        }
    }
}
```

```
user@A# show protocols
ospf {
    area 0.0.0.0 {
        interface fe-1/2/0.4;
    }
}
```

Device B

```
user@B# show interfaces
interfaces {
    fe-1/2/0 {
        unit 3 {
            description B->A;
            family inet {
                address 10.0.2.1/30;
            }
        }
    }
    fe-1/2/1 {
        unit 6 {
            description B->C;
            family inet {
                address 10.0.3.1/30;
            }
        }
    }
}
```

```
user@B# show policy-options
policy-statement send-static {
    from protocol static;
    then accept;
}
```

```
user@B# show protocols
bgp {
    group ext {
        type external;
        peer-as 23;
        neighbor 10.0.3.2;
    }
}
ospf {
    export send-static;
    area 0.0.0.0 {
        interface fe-1/2/0.3;
    }
}
```

```
user@B# show routing-options
static {
    route 0.0.0.0/0 next-hop 10.0.3.2;
    route 192.168.0.0/24 {
        next-hop 10.0.3.2;
        no-readvertise;
    }
}
autonomous-system 17;
```

Device C

```
user@C# show interfaces
fe-1/2/0 {
    unit 7 {
        description B->C;
        family inet {
            address 10.0.3.2/30;
        }
    }
}
lo0 {
    unit 5 {
        family inet {
            address 192.168.0.1/32;
        }
    }
}
```

```
user@C# show protocols
bgp {
    group ext {
        type external;
        peer-as 17;
        neighbor 10.0.3.1;
    }
}
```

```
user@C# show routing-options
autonomous-system 23;
```

If you are done configuring the devices, enter **commit** from configuration mode.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Checking the Routing Table

Purpose

Make sure that the `no-readvertise` statement is working.

Action

1. On Device A, run the `show route protocol ospf` command to make sure that the 192.168.0.0/24 route does not appear in Device A's routing table.

```
user@A> show route protocols ospf
inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[OSPF/150] 00:03:15, metric 0, tag 0
                    > to 10.0.2.1 via fe-1/2/0.4
224.0.0.5/32       *[OSPF/10] 00:04:07, metric 1
                      MultiRecv
```

2. On Device B, deactivate the `no-readvertise` statement.

```
user@B# deactivate routing-options static route 192.168.0.0/24 no-readvertise
```

3. On Device A, rerun the `show route protocol ospf` command to make sure that the 192.168.0.0/24 route appears in Device A's routing table.

```
user@A> show route protocols ospf
inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[OSPF/150] 00:04:24, metric 0, tag 0
                    > to 10.0.2.1 via fe-1/2/0.4
192.168.0.0/24     *[OSPF/150] 00:00:15, metric 0, tag 0
                    > to 10.0.2.1 via fe-1/2/0.4
224.0.0.5/32       *[OSPF/10] 00:05:16, metric 1
                      MultiRecv
```

Meaning

The `no-readvertise` statement is working as expected.

Verifying the Static Route Configuration

IN THIS SECTION

Purpose

Verify that the static routes are in the routing table and that those routes are active.

Action

From the CLI, enter the `show route terse` command.

Sample Output

command-name

```
user@host> show route terse
inet.0: 20 destinations, 20 routes (20 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 192.168.47.5/32        S   5                        Reject
* 172.16.0.0/12      S   5                       >192.168.71.254
* 192.168.0.0/18     S   5                       >192.168.71.254
* 192.168.40.0/22    S   5                       >192.168.71.254
* 192.168.64.0/18    S   5                       >192.168.71.254
* 192.168.64.0/21    D   0                       >fxp0.0
* 192.168.71.246/32  L   0                        Local
* 192.168.220.4/30   D   0                       >ge-0/0/1.0
* 192.168.220.5/32   L   0                        Local
* 192.168.220.8/30   D   0                       >ge-0/0/2.0
* 192.168.220.9/32   L   0                        Local
* 192.168.220.12/30  D   0                       >ge-0/0/3.0
* 192.168.220.13/32  L   0                        Local
* 192.168.220.17/32  L   0                        Reject
* 192.168.220.21/32  L   0                        Reject
* 192.168.220.24/30  D   0                       >at-1/0/0.0
* 192.168.220.25/32  L   0                        Local
* 192.168.220.28/30  D   0                       >at-1/0/1.0
* 192.168.220.29/32  L   0                        Local
* 224.0.0.9/32       R 100          1             MultiRecv
```

Meaning

The output shows a list of the routes that are currently in the **inet.0** routing table. Verify the following information:

- Each configured static route is present. Routes are listed in ascending order by IP address. Static routes are identified with an **S** in the protocol (**P**) column of the output.

- Each static route is active. Routes that are active show the next-hop IP address in the **Next hop** column. If a route's next-hop address is unreachable, the next-hop address is identified as **Reject**. These routes are not active routes, but they appear in the routing table because the **passive** attribute is set.
- The preference for each static route is correct. The preference for a particular route is listed in the **Prf** column of the output.

SEE ALSO

show route terse

CLI Explorer

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
17.1R1	This feature is supported on QFX10000 switches starting with Junos OS 17.1R1.

Bidirectional Forwarding Detection for Static Routes

IN THIS SECTION

Understanding BFD for Static Routes for Faster Network Failure Detection

The Bidirectional Forwarding Detection (BFD) protocol is a simple hello mechanism that detects failures in a network. BFD works with a wide variety of network environments and topologies. A pair of routing devices exchanges BFD packets. Hello packets are sent at a specified, regular interval. A neighbor failure is detected when the routing device stops receiving a reply after a specified interval. The BFD failure detection timers have shorter time limits than the static route failure detection mechanisms, so they provide faster detection.

The BFD failure detection timers can be adjusted to be faster or slower. The lower the BFD failure detection timer value, the faster the failure detection and vice versa. For example, the timers can adapt to a higher value if the adjacency fails (that is, the timer detects failures more slowly). Or a neighbor can negotiate a higher value for a timer than the configured value. The timers adapt to a higher value when a BFD session flap occurs more than three times in a span of 15 seconds. A back-off algorithm increases the receive (Rx) interval by two if the local BFD instance is the reason for the session flap. The transmission (Tx) interval is increased by two if the remote BFD instance is the reason for the session flap. You can use the `clear bfd adaptation` command to return BFD interval timers to their configured values. The `clear bfd adaptation` command is hitless, meaning that the command does not affect traffic flow on the routing device.

By default, BFD is supported on single-hop static routes.

NOTE: On MX Series devices, multihop BFD is not supported on a static route if the static route is configured with more than one next hop. It is recommended that you avoid using multiple next hops when a multihop BFD is required for a static route.

To enable failure detection, include the `bfd-liveness-detection` statement in the static route configuration.

NOTE: The `bfd-liveness-detection` command includes the description field. On devices that support this feature, the description is an attribute under the **bfd-liveness-detection** object. This field is applicable only for the static routes.

The BFD protocol is supported for IPv6 static routes. Global unicast and link-local IPv6 addresses are supported for static routes. The BFD protocol is not supported on multicast or anycast IPv6 addresses. For IPv6, the BFD protocol supports only static routes. IPv6 for BFD is also supported for the eBGP protocol.

To configure the BFD protocol for IPv6 static routes, include the `bfd-liveness-detection` statement at the `[edit routing-options rib inet6.0 static route` *`destination-prefix`*`]` hierarchy level.

You can configure a hold-down interval to specify how long the BFD session must remain up before a state change notification is sent.

To specify the hold-down interval, include the `holddown-interval` statement in the BFD configuration. You can configure a number in the range from 0 through 255,000 milliseconds. The default is 0. If the BFD session goes down and then comes back up during the hold-down interval, the timer is restarted.

> **NOTE**: If a single BFD session includes multiple static routes, the hold-down interval with the highest value is used.

To specify the minimum transmit and receive intervals for failure detection, include the `minimum-interval` statement in the BFD configuration.

This value represents both the minimum interval after which the local routing device transmits hello packets and the minimum interval after which the routing device expects to receive a reply from the neighbor with which it has established a BFD session. You can configure a number in the range from 1 through 255,000 milliseconds. Optionally, instead of using this statement, you can configure the minimum transmit and receive intervals separately using the **transmit-interval minimum-interval** and `minimum-receive-interval` statements.

> **NOTE**: Depending on your network environment, these additional recommendations might apply:
>
> - The recommended minimum interval for centralised BFD is 300 ms with a `multiplier` of 3, and the reccomended minimum interval for distributed BFD is 100 ms with a `multiplier` of 3.
> - For very large-scale network deployments with a large number of BFD sessions, contact Juniper Networks customer support for more information.
> - For BFD sessions to remain up during a Routing Engine switchover event when *nonstop active routing* (NSR) is configured, specify a minimum interval of 2500 ms for Routing Engine-based sessions. For distributed BFD sessions with NSR configured, the minimum interval recommendations are unchanged and depend only on your network deployment.

To specify the minimum receive interval for failure detection, include the `minimum-receive-interval` statement in the BFD configuration. This value represents the minimum interval after which the routing device expects to receive a reply from a neighbor with which it has established a BFD session. You can configure a number in the range from 1 through 255,000 milliseconds. Optionally, instead of using this statement, you can configure the minimum receive interval using the `minimum-interval` statement at the `[edit routing-options static route` *`destination-prefix`* `bfd-liveness-detection]` hierarchy level.

To specify the number of hello packets not received by the neighbor that causes the originating interface to be declared down, include the `multiplier` statement in the BFD configuration. The default value is 3. You can configure a number in the range from 1 through 255.

To specify a threshold for detecting the adaptation of the detection time, include the `threshold` statement in the BFD configuration.

When the BFD session detection time adapts to a value equal to or higher than the threshold, a single trap and a system log message are sent. The detection time is based on the multiplier of the **minimum-interval** or the **minimum-receive-interval** value. The threshold must be a higher value than the multiplier for either of these configured values. For example if the **minimum-receive-interval** is 300 ms and the **multiplier** is 3, the total detection time is 900 ms. Therefore, the detection time threshold must have a value higher than 900.

To specify the minimum transmit interval for failure detection, include the `transmit-interval minimum-interval` statement in the BFD configuration.

This value represents the minimum interval after which the local routing device transmits hello packets to the neighbor with which it has established a BFD session. You can configure a value in the range from 1 through 255,000 milliseconds. Optionally, instead of using this statement, you can configure the minimum transmit interval using the `minimum-interval` statement at the `[edit routing-options static route` ***`destination-prefix`*** `bfd-liveness-detection]` hierarchy level.

To specify the threshold for the adaptation of the transmit interval, include the `transmit-interval threshold` statement in the BFD configuration.

The threshold value must be greater than the transmit interval. When the BFD session transmit time adapts to a value greater than the threshold, a single trap and a system log message are sent. The detection time is based on the multiplier of the value for the **minimum-interval** or the `minimum-receive-interval` statement at the `[edit routing-options static route` ***`destination-prefix`*** `bfd-liveness-detection]` hierarchy level. The threshold must be a higher value than the multiplier for either of these configured values.

To specify the BFD version, include the `version` statement in the BFD configuration. The default is to have the version detected automatically.

To include an IP address for the next hop of the BFD session, include the `neighbor` statement in the BFD configuration.

NOTE: You must configure the `neighbor` statement if the next hop specified is an interface name. If you specify an IP address as the next hop, that address is used as the neighbor address for the BFD session.

You can configure BFD sessions not to adapt to changing network conditions. To disable BFD adaptation, include the `no-adaptation` statement in the BFD configuration.

NOTE: We recommend that you not disable BFD adaptation unless it is preferable *not* to have BFD adaptation in your network.

NOTE: If BFD is configured only on one end of a static route, the route is removed from the routing table. BFD establishes a session when BFD is configured on both ends of the static route.

BFD is not supported on ISO address families in static routes. BFD does support IS-IS.

If you configure *graceful Routing Engine switchover* (GRES) at the same time as BFD, GRES does not preserve the BFD state information during a failover.

Example: Configuring BFD for Static Routes for Faster Network Failure Detection

IN THIS SECTION

This example shows how to configure Bidirectional Forwarding Detection (BFD) for static routes.

Requirements

In this example, no special configuration beyond device initialization is required.

Overview

IN THIS SECTION

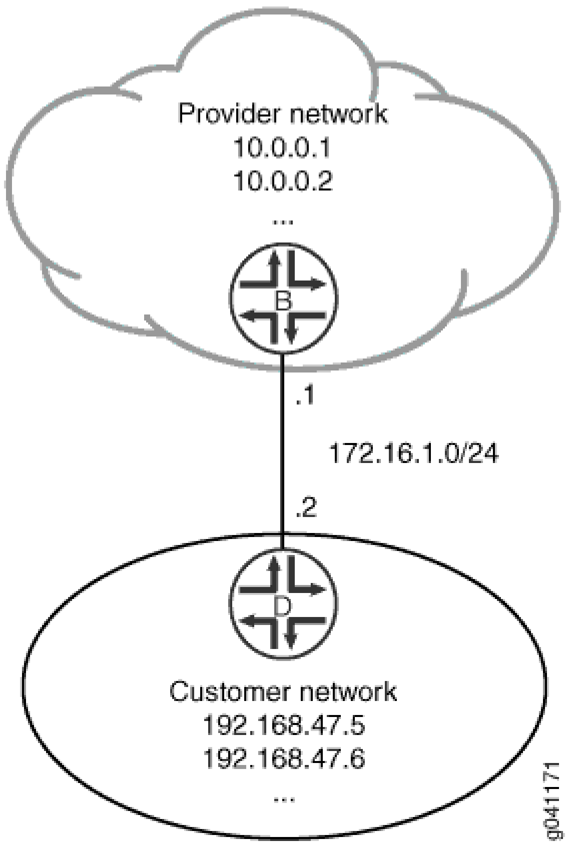
There are many practical applications for static routes. Static routing is often used at the network edge to support attachment to stub networks, which, given their single point of entry and egress, are well suited to the simplicity of a static route. In Junos OS, static routes have a global preference of 5. Static routes are activated if the specified next hop is reachable.

In this example, you configure the static route 192.168.47.0/24 from the provider network to the customer network, using the next-hop address of 172.16.1.2. You also configure a static default route of 0.0.0.0/0 from the customer network to the provider network, using a next-hop address of 172.16.1.1.

For demonstration purposes, some loopback interfaces are configured on Device B and Device D. These loopback interfaces provide addresses to ping and thus verify that the static routes are working.

Figure 7 on page 70 shows the sample network.

Figure 7: Customer Routes Connected to a Service Provider



Topology

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

Device B

```
set interfaces ge-1/2/0 unit 0 description B->D
set interfaces ge-1/2/0 unit 0 family inet address 172.16.1.1/24
set interfaces lo0 unit 57 family inet address 10.0.0.1/32
set interfaces lo0 unit 57 family inet address 10.0.0.2/32
set routing-options static route 192.168.47.0/24 next-hop 172.16.1.2
set routing-options static route 192.168.47.0/24 bfd-liveness-detection minimum-interval 1000
set routing-options static route 192.168.47.0/24 bfd-liveness-detection description Site-xxx
set protocols bfd traceoptions file bfd-trace
set protocols bfd traceoptions flag all
```

Device D

```
set interfaces ge-1/2/0 unit 1 description D->B
set interfaces ge-1/2/0 unit 1 family inet address 172.16.1.2/24
set interfaces lo0 unit 2 family inet address 192.168.47.5/32
set interfaces lo0 unit 2 family inet address 192.168.47.6/32
set routing-options static route 0.0.0.0/0 next-hop 172.16.1.1
set routing-options static route 0.0.0.0/0 bfd-liveness-detection minimum-interval 1000
set protocols bfd traceoptions file bfd-trace
set protocols bfd traceoptions flag all
```

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure BFD for static routes:

1. On Device B, configure the interfaces.

```
[edit interfaces]
user@B# set ge-1/2/0 unit 0 description B->D
user@B# set ge-1/2/0 unit 0 family inet address 172.16.1.1/24
user@B# set lo0 unit 57 family inet address 10.0.0.1/32
user@B# set lo0 unit 57 family inet address 10.0.0.2/32
```

2. On Device B, create a static route and set the next-hop address.

```
[edit routing-options]
user@B# set static route 192.168.47.0/24 next-hop 172.16.1.2
```

3. On Device B, configure BFD for the static route.

```
[edit routing-options]
user@B# set static route 192.168.47.0/24 bfd-liveness-detection minimum-interval 1000
set routing-options static route 192.168.47.0/24 bfd-liveness-detection description Site-
xxx
```

4. On Device B, configure tracing operations for BFD.

```
[edit protocols]
user@B# set bfd traceoptions file bfd-trace
user@B# set bfd traceoptions flag all
```

5. If you are done configuring Device B, commit the configuration.

```
[edit]
user@B# commit
```

6. On Device D, configure the interfaces.

```
[edit interfaces]
user@D# set ge-1/2/0 unit 1 description D->B
user@D# set ge-1/2/0 unit 1 family inet address 172.16.1.2/24
user@D# set lo0 unit 2 family inet address 192.168.47.5/32
user@D# set lo0 unit 2 family inet address 192.168.47.6/32
```

7. On Device D, create a static route and set the next-hop address.

```
[edit routing-options]
user@D# set static route 0.0.0.0/0 next-hop 172.16.1.1
```

8. On Device D, configure BFD for the static route.

```
[edit routing-options]
user@D# set static route 0.0.0.0/0 bfd-liveness-detection minimum-interval 1000
```

9. On Device D, configure tracing operations for BFD.

```
[edit protocols]
user@D# set bfd traceoptions file bfd-trace
user@D# set bfd traceoptions flag all
```

10. If you are done configuring Device D, commit the configuration.

```
[edit]
user@D# commit
```

Results

Confirm your configuration by issuing the `show interfaces`, `show protocols`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device B

```
user@B# show interfaces
ge-1/2/0 {
    unit 0 {
        description B->D;
        family inet {
            address 172.16.1.1/24;
        }
    }
}
lo0 {
    unit 57 {
        family inet {
            address 10.0.0.1/32;
            address 10.0.0.2/32;
        }
    }
}
```

```
user@D# show protocols
bfd {
    traceoptions {
        file bfd-trace;
        flag all;
    }
}
```

```
user@B# show routing-options
static {
    route 192.168.47.0/24 {
        next-hop 172.16.1.2;
        bfd-liveness-detection {
            description Site- xxx;
            minimum-interval 1000;
        }
    }
}
```

Device D

```
user@D# show interfaces
ge-1/2/0 {
    unit 1 {
        description D->B;
        family inet {
            address 172.16.1.2/24;
        }
    }
}
lo0 {
    unit 2 {
        family inet {
            address 192.168.47.5/32;
            address 192.168.47.6/32;
        }
    }
}
```

```
user@D# show routing-options
static {
    route 0.0.0.0/0 {
        next-hop 172.16.1.1;
        bfd-liveness-detection {
            description Site - xxx;
            minimum-interval 1000;
        }
    }
}
```

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying That BFD Sessions Are Up

Purpose

Verify that the BFD sessions are up, and view details about the BFD sessions.

Action

From operational mode, enter the `show bfd session extensive` command.

```
user@B> show bfd session extensive
                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
172.16.1.2               Up        lt-1/2/0.0     3.000     1.000        3
 Client Static, description Site-xxx, TX interval 1.000, RX interval 1.000
 Session up time 00:14:30
 Local diagnostic None, remote diagnostic None
 Remote state Up, version 1
 Replicated, routing table index 172
 Min async interval 1.000, min slow interval 1.000
 Adaptive async TX interval 1.000, RX interval 1.000
 Local min TX interval 1.000, minimum RX interval 1.000, multiplier 3
 Remote min TX interval 1.000, min RX interval 1.000, multiplier 3
 Local discriminator 2, remote discriminator 1
 Echo mode disabled/inactive

1 sessions, 1 clients
Cumulative transmit rate 1.0 pps, cumulative receive rate 1.0 pps
```

NOTE: The **description Site- <*xxx*>** is supported only on the SRX Series Firewalls.

If each client has more than one description field, then it displays "and more" along with the first description field.

```
user@D> show bfd session extensive
                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
172.16.1.1               Up        lt-1/2/0.1     3.000     1.000        3
 Client Static, TX interval 1.000, RX interval 1.000
 Session up time 00:14:35
 Local diagnostic None, remote diagnostic None
 Remote state Up, version 1
 Replicated, routing table index 170
 Min async interval 1.000, min slow interval 1.000
 Adaptive async TX interval 1.000, RX interval 1.000
 Local min TX interval 1.000, minimum RX interval 1.000, multiplier 3
 Remote min TX interval 1.000, min RX interval 1.000, multiplier 3
 Local discriminator 1, remote discriminator 2
 Echo mode disabled/inactive

1 sessions, 1 clients
Cumulative transmit rate 1.0 pps, cumulative receive rate 1.0 pps
```

Meaning

The `TX interval 1.000, RX interval 1.000` output represents the setting configured with the `minimum-interval` statement. All of the other output represents the default settings for BFD. To modify the default settings, include the optional statements under the `bfd-liveness-detection` statement.

Viewing Detailed BFD Events

Purpose

View the contents of the BFD trace file to assist in troubleshooting, if needed.

Action

From operational mode, enter the `file show /var/log/bfd-trace` command.

```
user@B> file show /var/log/bfd-trace
Nov 23 14:26:55    Data (9) len 35: (hex) 42 46 44 20 70 65 72 69 6f 64 69 63 20 78 6d 69 74 20
72
Nov 23 14:26:55 PPM Trace: BFD periodic xmit rt tbl index 172
Nov 23 14:26:55 Received Downstream TraceMsg (22) len 108:
Nov 23 14:26:55    IfIndex (3) len 4: 0
Nov 23 14:26:55    Protocol (1) len 1: BFD
Nov 23 14:26:55    Data (9) len 83: (hex) 70 70 6d 64 5f 62 66 64 5f 73 65 6e 64 6d 73 67 20 3a
20
Nov 23 14:26:55 PPM Trace: ppmd_bfd_sendmsg : socket 12 len 24, ifl 78 src 172.16.1.1 dst
172.16.1.2 errno 65
Nov 23 14:26:55 Received Downstream TraceMsg (22) len 93:
Nov 23 14:26:55    IfIndex (3) len 4: 0
Nov 23 14:26:55    Protocol (1) len 1: BFD
Nov 23 14:26:55    Data (9) len 68: (hex) 42 46 44 20 70 65 72 69 6f 64 69 63 20 78 6d 69 74 20
74
```

Meaning

BFD messages are being written to the trace file.

Understanding BFD Authentication for Static Route Security

IN THIS SECTION

Bidirectional Forwarding Detection (BFD) enables rapid detection of communication failures between adjacent systems. By default, authentication for BFD sessions is disabled. However, when you run BFD over Network Layer protocols, the risk of service attacks can be significant.

NOTE: We strongly recommend using authentication if you are running BFD over multiple hops or through insecure tunnels.

Beginning with Junos OS Release 9.6, Junos OS supports authentication for BFD sessions running over IPv4 and IPv6 static routes. BFD authentication is not supported on MPLS OAM sessions. BFD authentication is only supported in the Canada and United States version of the Junos OS image and is not available in the export version.

NOTE: EX3300 supports BFD over static routes only.

You authenticate BFD sessions by specifying an authentication algorithm and keychain, and then associating that configuration information with a security authentication keychain using the keychain name.

The following sections describe the supported authentication algorithms, security keychains, and level of authentication that can be configured:

BFD Authentication Algorithms

Junos OS supports the following algorithms for BFD authentication:

- **simple-password**—Plain-text password. One to 16 bytes of plain text are used to authenticate the BFD session. One or more passwords can be configured. This method is the least secure and should be used only when BFD sessions are not subject to packet interception.

- **keyed-md5**—Keyed Message Digest 5 hash algorithm for sessions with transmit and receive intervals greater than 100 ms. To authenticate the BFD session, keyed MD5 uses one or more secret keys (generated by the algorithm) and a sequence number that is updated periodically. With this method, packets are accepted at the receiving end of the session if one of the keys matches and the sequence number is greater than or equal to the last sequence number received. Although more secure than a simple password, this method is vulnerable to replay attacks. Increasing the rate at which the sequence number is updated can reduce this risk.

- **meticulous-keyed-md5**—Meticulous keyed Message Digest 5 hash algorithm. This method works in the same manner as keyed MD5, but the sequence number is updated with every packet. Although more secure than keyed MD5 and simple passwords, this method might take additional time to authenticate the session.

- **keyed-sha-1**—Keyed Secure Hash Algorithm I for sessions with transmit and receive intervals greater than 100 ms. To authenticate the BFD session, keyed SHA uses one or more secret keys (generated by the algorithm) and a sequence number that is updated periodically. The key is not carried within the packets. With this method, packets are accepted at the receiving end of the session if one of the keys matches and the sequence number is greater than the last sequence number received.

- **meticulous-keyed-sha-1**—Meticulous keyed Secure Hash Algorithm I. This method works in the same manner as keyed SHA, but the sequence number is updated with every packet. Although more secure than keyed SHA and simple passwords, this method might take additional time to authenticate the session.

NOTE: *Nonstop active routing* (NSR) is not supported with meticulous-keyed-md5 and meticulous-keyed-sha-1 authentication algorithms. BFD sessions using these algorithms might go down after a switchover.

NOTE: QFX5000 Series switches and EX4600 switches do not support minimum interval values of less than 1 second.

Security Authentication Keychains

The security authentication keychain defines the authentication attributes used for authentication key updates. When the security authentication keychain is configured and associated with a protocol through the keychain name, authentication key updates can occur without interrupting routing and signaling protocols.

The authentication keychain contains one or more keychains. Each keychain contains one or more keys. Each key holds the secret data and the time at which the key becomes valid. The algorithm and keychain must be configured on both ends of the BFD session, and they must match. Any mismatch in configuration prevents the BFD session from being created.

BFD allows multiple clients per session, and each client can have its own keychain and algorithm defined. To avoid confusion, we recommend specifying only one security authentication keychain.

Strict Versus Loose Authentication

By default, strict authentication is enabled, and authentication is checked at both ends of each BFD session. Optionally, to smooth migration from nonauthenticated sessions to authenticated sessions, you can configure *loose checking*. When loose checking is configured, packets are accepted without authentication being checked at each end of the session. This feature is intended for transitional periods only.

Example: Configuring BFD Authentication for Securing Static Routes

IN THIS SECTION

This example shows how to configure Bidirectional Forwarding Detection (BFD) authentication for static routes.

Requirements

Junos OS Release 9.6 or later (Canda and United States version).

BFD authentication is only supported in the Canada and United States version of the Junos OS image and is not available in the export version.

Overview

IN THIS SECTION

You can configure authentication for BFD sessions running over IPv4 and IPv6 static routes. Routing instances and logical systems are also supported.

The following steps are needed to configure authentication on a BFD session:

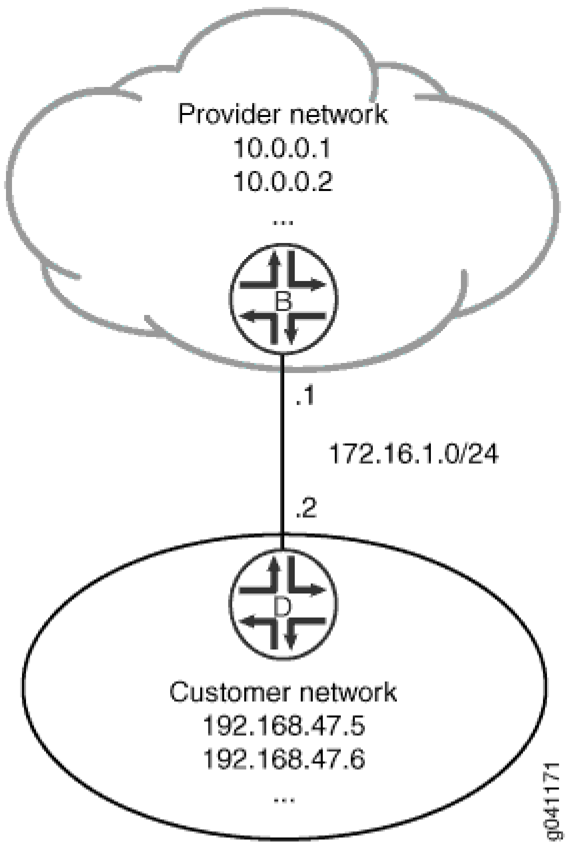
1. Specify the BFD authentication algorithm for the static route.
2. Associate the authentication keychain with the static route.
3. Configure the related security authentication keychain. This must be configured on the main router.

TIP: We recommend that you specify loose authentication checking if you are transitioning from nonauthenticated sessions to authenticated sessions.

```
[edit]
user@host> set routing-options static route ipv4 bfd-liveness-detection
authentication loose-check
```

Figure 8 on page 82 shows the sample network.

Figure 8: Customer Routes Connected to a Service Provider



Topology

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

Device B

```
set interfaces ge-1/2/0 unit 0 description B->D
set interfaces ge-1/2/0 unit 0 family inet address 172.16.1.1/24
set interfaces lo0 unit 57 family inet address 10.0.0.1/32
set interfaces lo0 unit 57 family inet address 10.0.0.2/32
set routing-options static route 192.168.47.0/24 next-hop 172.16.1.2
set routing-options static route 192.168.47.0/24 bfd-liveness-detection minimum-interval 1000
set routing-options static route 192.168.47.0/24 bfd-liveness-detection description Site-xxx
set routing-options static route 192.168.47.0/24 bfd-liveness-detection authentication key-chain
bfd-kc4
set routing-options static route 192.168.47.0/24 bfd-liveness-detection authentication algorithm
keyed-sha-1
set security authentication-key-chains key-chain bfd-kc4 key 5 secret "$ABC123$ABC123$ABC123"
set security authentication-key-chains key-chain bfd-kc4 key 5 start-time "2011-1-1.12:00:00
-0800"
```

Device D

```
set interfaces ge-1/2/0 unit 1 description D->B
set interfaces ge-1/2/0 unit 1 family inet address 172.16.1.2/24
set interfaces lo0 unit 2 family inet address 192.168.47.5/32
set interfaces lo0 unit 2 family inet address 192.168.47.6/32
set routing-options static route 0.0.0.0/0 next-hop 172.16.1.1
set routing-options static route 0.0.0.0/0 bfd-liveness-detection minimum-interval 1000
set routing-options static route 0.0.0.0/0 bfd-liveness-detection authentication key-chain bfd-
kc4
set routing-options static route 0.0.0.0/0 bfd-liveness-detection authentication algorithm keyed-
sha-1
set security authentication-key-chains key-chain bfd-kc4 key 5 secret "$ABC123$ABC123$ABC123"
set security authentication-key-chains key-chain bfd-kc4 key 5 start-time "2011-1-1.12:00:00
-0800"
```

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure BFD for static routes:

1. On Device B, configure the interfaces.

```
[edit interfaces]
user@B# set ge-1/2/0 unit 0 description B->D
user@B# set ge-1/2/0 unit 0 family inet address 172.16.1.1/24
user@B# set lo0 unit 57 family inet address 10.0.0.1/32
user@B# set lo0 unit 57 family inet address 10.0.0.2/32
```

2. On Device B, create a static route and set the next-hop address.

```
[edit routing-options]
user@B# set static route 192.168.47.0/24 next-hop 172.16.1.2
```

3. On Device B, configure BFD for the static route.

```
[edit routing-options]
user@B# set static route 192.168.47.0/24 bfd-liveness-detection minimum-interval 1000
set routing-options static route 192.168.47.0/24 bfd-liveness-detection description Site-xxx
```

4. On Device B, specify the algorithm (**keyed-md5**, **keyed-sha-1**, **meticulous-keyed-md5**, **meticulous-keyed-sha-1**, or **simple-password**) to use for BFD authentication on the static route.

```
[edit routing-options]
user@B# set static route 192.168.47.0/24 bfd-liveness-detection authentication algorithm
keyed-sha-1
```

NOTE: Nonstop active routing (NSR) is not supported with the meticulous-keyed-md5 and meticulous-keyed-sha-1 authentication algorithms. BFD sessions using these algorithms might go down after a switchover.

5. On Device B, specify the keychain to be used to associate BFD sessions on the specified route with the unique security authentication keychain attributes.

   This should match the keychain name configured at the `[edit security authentication key-chains]` hierarchy level.

```
[edit routing-options]
user@B# set static route 192.168.47.0/24 bfd-liveness-detection authentication key-chain bfd-
kc4
```

6. On Device B, specify the unique security authentication information for BFD sessions:

   - The matching keychain name as specified in Step 5.
   - At least one key, a unique integer between **0** and **63**. Creating multiple keys allows multiple clients to use the BFD session.
   - The secret data used to allow access to the session.
   - The time at which the authentication key becomes active, in the format *yyyy-mm-dd.hh:mm:ss*.

```
[edit security authentication-key-chains key-chain bfd-kc4]
user@B# set key 5 secret "$ABC123$ABC123$ABC123"
user@B# set key 5 start-time "2011-1-1.12:00:00 -0800"
```

7. If you are done configuring Device B, commit the configuration.

```
[edit]
user@B# commit
```

8. Repeat the configuration on Device D.

   The algorithm and keychain must be configured on both ends of the BFD session, and they must match. Any mismatch in configuration prevents the BFD session from being created.

Results

Confirm your configuration by issuing the `show interfaces`, `show routing-options`, and `show security` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device B

```
user@B# show interfaces
ge-1/2/0 {
    unit 0 {
        description B->D;
        family inet {
            address 172.16.1.1/24;
        }
    }
}
lo0 {
    unit 57 {
        family inet {
            address 10.0.0.1/32;
            address 10.0.0.2/32;
        }
    }
}
```

```
user@B# show routing-options
static {
    route 192.168.47.0/24 {
        next-hop 172.16.1.2;
        bfd-liveness-detection {
            description Site- xxx;
            minimum-interval 1000;
            authentication {
                key-chain bfd-kc4;
                algorithm keyed-sha-1;
            }
        }
    }
}
```

```
user@B# show security
authentication-key-chains {
    key-chain bfd-kc4 {
        key 5 {
            secret "$ABC123$ABC123$ABC123"; ## SECRET-DATA
            start-time "2011-1-1.12:00:00 -0800";
        }
    }
}
```

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying That BFD Sessions Are Up

Purpose

Verify that the BFD sessions are up.

Action

From operational mode, enter the `show bfd session` command.

```
user@B> show bfd session
                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
172.16.1.2               Up        ge-1/2/0.0     3.000     1.000       3

1 sessions, 1 clients
Cumulative transmit rate 1.0 pps, cumulative receive rate 1.0 pps
```

Meaning

The command output shows that the BFD session is up.

Viewing Details About the BFD Session

Purpose

View details about the BFD sessions and make sure that authentication is configured.

Action

From operational mode, enter the `show bfd session detail` command.

```
user@B> show bfd session detail
                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
172.16.1.2               Up        ge-1/2/0.0     3.000     1.000       3
 Client Static, TX interval 1.000, RX interval 1.000, Authenticate
 Session up time 00:53:58
 Local diagnostic NbrSignal, remote diagnostic None
 Remote state Up, version 1
 Logical system 9, routing table index 22

1 sessions, 1 clients
Cumulative transmit rate 1.0 pps, cumulative receive rate 1.0 pps
```

Meaning

In the command output, **Authenticate** is displayed to indicate that BFD authentication is configured.

Viewing Extensive BFD Session Information

Purpose

View more detailed information about the BFD sessions.

Action

From operational mode, enter the `show bfd session extensive` command.

```
user@B> show bfd session extensive
Address                  State     Interface      Time     Interval  Multiplier
172.16.1.2               Up        ge-1/2/0.0     3.000     1.000       3
 Client Static, description Site-xxx, TX interval 1.000, RX interval 1.000, Authenticate
        keychain bfd-kc4, algo keyed-sha-1, mode strict
 Session up time 01:39:45
 Local diagnostic NbrSignal, remote diagnostic None
 Remote state Up, version 1
 Logical system 9, routing table index 22
 Min async interval 1.000, min slow interval 1.000
 Adaptive async TX interval 1.000, RX interval 1.000
 Local min TX interval 1.000, minimum RX interval 1.000, multiplier 3
 Remote min TX interval 1.000, min RX interval 1.000, multiplier 3
 Local discriminator 3, remote discriminator 4
 Echo mode disabled/inactive
 Authentication enabled/active, keychain bfd-kc4, algo keyed-sha-1, mode strict

1 sessions, 1 clients
Cumulative transmit rate 1.0 pps, cumulative receive rate 1.0 pps
```

Meaning

In the command output, **Authenticate** is displayed to indicate that BFD authentication is configured. The output for the `extensive` command provides the keychain name, the authentication algorithm, and the mode for each client in the session.

NOTE: The **description Site- <*xxx*>** is supported only on the SRX Series Firewalls.

If each client has more than one description field, then it displays "and more" along with the first description field.

Example: Enabling BFD on Qualified Next Hops in Static Routes for Route Selection

IN THIS SECTION

This example shows how to configure a static route with multiple possible next hops. Each next hop has Bidirectional Forwarding Detection (BFD) enabled.

Requirements

In this example, no special configuration beyond device initialization is required.

Overview

IN THIS SECTION

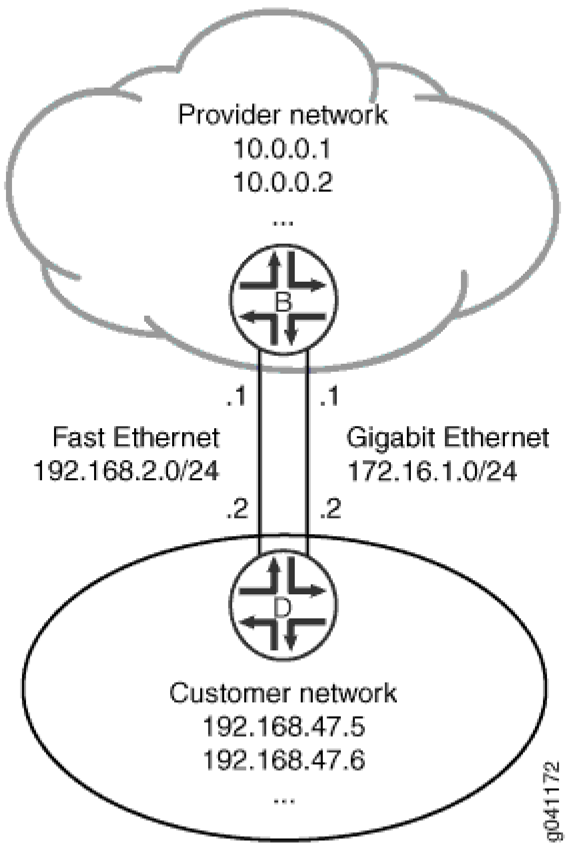
In this example, Device B has the static route **192.168.47.0/24** with two possible next hops. The two next hops are defined using two `qualified-next-hop` statements. Each next hop has BFD enabled.

BFD is also enabled on Device D because BFD must be enabled on both ends of the connection.

A next hop is included in the routing table if the BFD session is up. The next hop is removed from the routing table if the BFD session is down.

See Figure 9 on page 91.

Figure 9: BFD Enabled on Qualified Next Hops



Topology

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

Device B

```
set interfaces fe-0/1/0 unit 2 description secondary-B->D
set interfaces fe-0/1/0 unit 2 family inet address 192.168.2.1/24
set interfaces ge-1/2/0 unit 0 description B->D
set interfaces ge-1/2/0 unit 0 family inet address 172.16.1.1/24
set routing-options static route 192.168.47.0/24 qualified-next-hop 192.168.2.2 bfd-liveness-
detection minimum-interval 60
set routing-options static route 192.168.47.0/24 qualified-next-hop 172.16.1.2 bfd-liveness-
detection minimum-interval 60
```

Device D

```
set interfaces fe-0/1/0 unit 3 description secondary-D->B
set interfaces fe-0/1/0 unit 3 family inet address 192.168.2.2/24
set interfaces ge-1/2/0 unit 1 description D->B
set interfaces ge-1/2/0 unit 1 family inet address 172.16.1.2/24
set routing-options static route 0.0.0.0/0 qualified-next-hop 192.168.2.1
set routing-options static route 0.0.0.0/0 qualified-next-hop 172.16.1.1
set routing-options static route 0.0.0.0/0 bfd-liveness-detection minimum-interval 60
```

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure a static route with two possible next hops, both with BFD enabled:

1. On Device B, configure the interfaces.

   ```
   [edit interfaces fe-0/1/0]
   user@B# set unit 2 description secondary-B->D
   ```

```
user@B# set unit 2 family inet address 192.168.2.1/24
[edit interfaces ge-1/2/0]
user@B# set unit 0 description B->D
user@B# set unit 0 family inet address 172.16.1.1/24
```

2. On Device B, configure the static route with two next hops, both with BFD enabled.

```
[edit routing-options static route 192.168.47.0/24]
user@B# set qualified-next-hop 192.168.2.2 bfd-liveness-detection minimum-interval 60
user@B# set qualified-next-hop 172.16.1.2 bfd-liveness-detection minimum-interval 60
```

3. On Device D, configure the interfaces.

```
[edit interfaces fe-0/1/0]
user@D# set unit 3 description secondary-D->B
user@D# set unit 3 family inet address 192.168.2.2/24
[edit interfaces ge-1/2/0]
user@D# set unit 1 description D->B
user@D# set unit 1 family inet address 172.16.1.2/24
```

4. On Device D, configure a BFD-enabled default static route with two next hops to the provider network.

   In this case, BFD is enabled on the route, not on the next hops.

```
[edit routing-options static route 0.0.0.0/0]
user@D# set qualified-next-hop 192.168.2.1
user@D# set qualified-next-hop 172.16.1.1
user@D# set bfd-liveness-detection minimum-interval 60
```

Results

Confirm your configuration by issuing the `show interfaces` and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@B# show interfaces
fe-0/1/0 {
    unit 2 {
        description secondary-B->D;
        family inet {
            address 192.168.2.1/24;
        }
    }
}
ge-1/2/0 {
    unit 0 {
        description B->D;
        family inet {
            address 172.16.1.1/24;
        }
    }
}
```

```
user@B# show routing-options
static {
    route 192.168.47.0/24 {
        qualified-next-hop 192.168.2.2 {
            bfd-liveness-detection {
                minimum-interval 60;
            }
        }
        qualified-next-hop 172.16.1.2 {
            bfd-liveness-detection {
                minimum-interval 60;
            }
        }
    }
}
```

```
user@D# show interfaces
fe-0/1/0 {
    unit 3 {
        description secondary-D->B;
        family inet {
            address 192.168.2.2/24;
        }
    }
}
```

```
ge-1/2/0 {
    unit 1 {
        description D->B;
        family inet {
            address 172.16.1.2/24;
        }
    }
}
```

```
user@D# show routing-options
static {
    route 0.0.0.0/0 {
        qualified-next-hop 192.168.2.1;
        qualified-next-hop 172.16.1.1;
        bfd-liveness-detection {
            minimum-interval 60;
        }
    }
}
```

If you are done configuring the devices, enter **commit** from configuration mode.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Checking the Routing Tables

Purpose

Make sure that the static route appears in the routing table on Device B with two possible next hops.

Action

```
user@B> show route 192.168.47.0 extensive
inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
192.168.47.0/24 (1 entry, 1 announced)
TSI:
KRT in-kernel 192.168.47.0/24 -> {192.168.2.2}
        *Static Preference: 5
                Next hop type: Router
                Address: 0x9334010
                Next-hop reference count: 1
                Next hop: 172.16.1.2 via ge-1/2/0.0
                Next hop: 192.168.2.2 via fe-0/1/0.2, selected
                State: <Active Int Ext>
                Age: 9
                Task: RT
                Announcement bits (1): 3-KRT
                AS path: I
```

Meaning

Both next hops are listed. The next hop 192.168.2.2 is the selected route.

Verifying the BFD Sessions

Purpose

Make sure that the BFD sessions are up.

Action

```
user@B> show bfd session

                                          Detect   Transmit
```

```
Address          State     Interface      Time      Interval  Multiplier
172.16.1.2       Up        ge-1/2/0.0     0.720      0.240        3
192.168.2.2      Up        fe-0/1/0.2     0.720      0.240        3

2 sessions, 2 clients
Cumulative transmit rate 8.3 pps, cumulative receive rate 8.3 pps
```

Meaning

The output shows that the BFD sessions are up.

Removing BFD from Device D

Purpose

Demonstrate what happens when the BFD session is down for both next hops.

Action

1. Deactivate BFD on Device D.

```
[edit routing-options static route 0.0.0.0/0]
user@D# deactivate bfd-liveness-detection
user@D# commit
```

2. Rerun the `show bfd session` command on Device B.

```
user@B> show bfd session

                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
172.16.1.2               Down      ge-1/2/0.0     3.000     1.000        3
192.168.2.2              Down      fe-0/1/0.2     3.000     1.000        3

2 sessions, 2 clients
Cumulative transmit rate 2.0 pps, cumulative receive rate 2.0 pps
```

3. Rerun the `show route 192.168.47.0` command on Device B.

```
user@B> show route 192.168.47.0
```

Meaning

As expected, when the BFD sessions are down, the static route is removed from the routing table.

Removing BFD from One Next Hop

Purpose

Demonstrate what happens when only one next hop has BFD enabled.

Action

1. If it is not already deactivated, deactivate BFD on Device D.

```
[edit routing-options static route 0.0.0.0/0]
user@D# deactivate bfd-liveness-detection
user@D# commit
```

2. Deactivate BFD on one of the next hops on Device B.

```
[edit routing-options static route 192.168.47.0/24 qualified-next-hop 172.16.1.2]
user@B# deactivate bfd-liveness-detection
user@B# commit
```

3. Rerun the `show bfd session` command on Device B.

```
user@B> show bfd session

                                        Detect   Transmit
Address      State     Interface        Time     Interval  Multiplier
192.168.2.2   Down      fe-0/1/0.2       3.000     1.000        3
```

4. Rerun the `show route 192.168.47.0 extensive` command on Device B.

```
user@B> show route 192.168.47.0 extensive

inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
192.168.47.0/24 (1 entry, 1 announced)
TSI:
KRT in-kernel 192.168.47.0/24 -> {172.16.1.2}
        *Static Preference: 5
                Next hop type: Router, Next hop index: 624
                Address: 0x92f0178
                Next-hop reference count: 3
                Next hop: 172.16.1.2 via ge-1/2/0.0, selected
                State: <Active Int Ext>
                Age: 2:36
                Task: RT
                Announcement bits (1): 3-KRT
                AS path: I
```

Meaning

As expected, the BFD session is down for the 192.168.2.2 next hop. The 172.16.1.2 next hop remains in the routing table, and the route remains active, because BFD is not a condition for this next hop to remain valid.

Static Routes for CLNS

IN THIS SECTION

Understanding Static Routes for CLNS

The Connectionless Network Service (CLNS) is an ISO Layer 3 protocol that uses network service access point (NSAP) reachability information instead of IPv4 or IPv6 prefixes.

You can configure static routes to exchange CLNS routes within a CLNS island. A *CLNS island* is typically an IS-IS level 1 area that is part of a single IGP routing domain. An island can contain more than one area. CLNS islands can be connected by VPNs.

Example: Configuring Static Routes for CLNS When No IGP is Present

IN THIS SECTION

This example shows how to configure static routes for CLNS.

Requirements

Before you begin, configure the network interfaces. See Interfaces User Guide for Security Devices.

Overview

In this example, you configure static routes for CLNS. In the absence of an interior gateway protocol (IGP) on a certain link, a routing device might need to be configured with static routes for CLNS prefixes to be reachable by way of that link. This might be useful, for example, at an autonomous system (AS) boundary.

When you configure static routes for CLNS, consider the following tasks:

- Specify the `iso.0` routing table option to configure a primary instance CLNS static route.
- Specify the `instance-name.iso.0` routing table option to configure a CLNS static route for a particular routing instance.
- Specify the `route` *`nsap-prefix`* statement to configure the destination for the CLNS static route.

- Specify the `next-hop (interface-name | iso-net)` statement to configure the next hop, specified as an ISO network entity title (NET) or interface name.
- Include the `qualified-next-hop (interface-name | iso-net)` statement to configure a secondary backup next hop, specified as an ISO network entity title or interface name.

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set routing-options rib iso.0 static iso-route 47.0005.80ff.f800.0000.ffff.ffff/152 next-hop
47.0005.80ff.f800.0000.0108.0001.1921.6800.4212
set routing-options rib iso.0 static iso-route
47.0005.80ff.f800.0000.0108.0001.1921.6800.4212/152 next-hop t1-0/2/2.0
set routing-options rib iso.0 static iso-route 47.0005.80ff.f800.0000.eee0/152 qualified-next-
hop 47.0005.80ff.f800.0000.0108.0001.1921.6800.4002 preference 20
set routing-options rib iso.0 static iso-route 47.0005.80ff.f800.0000.eee0/152 qualified-next-
hop 47.0005.80ff.f800.0000.0108.0001.1921.6800.4002 metric 10
```

Procedure

Step-by-Step Procedure

To configure static routes for CLNS:

1. Configure the routes.

```
[edit routing-options rib iso.0 static]
user@host# set iso-route 47.0005.80ff.f800.0000.ffff.ffff/152 next-hop
```

```
47.0005.80ff.f800.0000.0108.0001.1921.6800.4212
user@host# set iso-route 47.0005.80ff.f800.0000.0108.0001.1921.6800.4212/152 next-hop
t1-0/2/2.0
user@host# set iso-route 47.0005.80ff.f800.0000.eee0/152 qualified-next-hop
47.0005.80ff.f800.0000.0108.0001.1921.6800.4002 preference 20
user@host# set iso-route 47.0005.80ff.f800.0000.eee0/152 qualified-next-hop
47.0005.80ff.f800.0000.0108.0001.1921.6800.4002 metric 10
```

2. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Results

Confirm your configuration by issuing the `show routing-options` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show routing-options
rib iso.0 {
    static {
        iso-route 47.0005.80ff.f800.0000.ffff.ffff/152 next-hop
47.0005.80ff.f800.0000.0108.0001.1921.6800.4212;
        iso-route 47.0005.80ff.f800.0000.0108.0001.1921.6800.4212/152 next-hop t1-0/2/2.0;
        iso-route 47.0005.80ff.f800.0000.eee0/152 {
            qualified-next-hop 47.0005.80ff.f800.0000.0108.0001.1921.6800.4002 {
                preference 20;
                metric 10;
            }
        }
    }
}
```

Verification

IN THIS SECTION

Checking the Routing Table

Purpose

Make sure that the expected routes appear in the routing table.

Action

```
user@host> show route table iso.0

iso.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

47.0005.80ff.f800.0000.0108.0001.1921.6800.4212/152
                   *[Static/5] 00:00:25
                    > via t1-0/2/2.0
47.0005.80ff.f800.0000.eee0/84
                   *[Static/20] 00:04:01, metric 10, metric2 10
                    > to #75 0.12.0.34.0.56 via fe-0/0/1.0
47.0005.80ff.f800.0000.ffff.ffff/104
                   *[Static/5] 00:04:01, metric2 0
                    > via t1-0/2/2.0
```

Meaning

The static routes appear in the routing table.

RELATED DOCUMENTATION

CLNS Configuration Overview

4

CHAPTER

Route Aggregation

IN THIS CHAPTER

Configuring Route Aggregation

IN THIS SECTION

Understanding Route Aggregation

IN THIS SECTION

The route aggregation methodology helps minimize the number of routing entries in an IP network by consolidating selected multiple routes into a single route advertisement. This approach is in contrast to non-aggregation routing, in which every routing table contains a unique entry for each route. The aggregation methodology does not help reduce the size of the routing-table on the router that does the aggregation. When you configure an export policy that only advertises the aggregate but not the contributing routes anymore, you then have the aggregation effect on the routers that receive updates.

An aggregate route becomes active when it has one or more *contributing routes*. A contributing route is an active route that is a more specific match for the aggregate destination. For example, for the aggregate destination `192.168.0.0/16`, routes to `192.168.192.0/19` and `192.168.67.0/24` are contributing routes, but routes to `192.168.0.0./8` and `192.168.0.0/16` are not.

A route can only contribute to a single aggregate route. However, an active aggregate route can recursively contribute to a less-specific matching aggregate route. For example, an aggregate route to the destination `192.168.0.0/16` can contribute to an aggregate route to `192.168.0.0/13`.

When an aggregate route becomes active, it is installed in the routing table with the following information:

- Reject next hop—If a more-specific packet does not match a more-specific route, the packet is rejected and an ICMP unreachable message is sent to the packet's originator.
- Metric value as configured with the `aggregate` statement.
- Preference value that results from the policy filter on the primary contributor, if a filter is specified.
- AS path as configured in the `aggregate` statement, if any. Otherwise, the path is computed by aggregating the paths of all contributing routes.
- Community as configured in the `aggregate` statement, if any is specified.

NOTE: You can configure only one aggregate route for each destination prefix.

To configure aggregate routes in the default routing table (`inet.0`), include the `aggregate` statement:

```
aggregate {
    defaults {
        ... aggregate-options ...
    }
    route destination-prefix {
        policy policy-name;
        ... aggregate-options ...
    }
}
```

To configure aggregate routes in one of the other routing tables, or to explicitly configure aggregate routes in the default routing table (`inet.0`), include the `aggregate` statement:

```
rib routing-table-name {
    aggregate {
        defaults {
            ... aggregate-options ...
        }
        route destination-prefix {
            policy policy-name;
            ... aggregate-options ...
        }
    }
}
```

NOTE: You cannot configure aggregate routes for the IPv4 multicast routing table (inet.1) nor the IPv6 multicast routing table (inet6.1).

The `aggregate` statement consists of two parts:

- `defaults`—(Optional) Here you specify global aggregate route options. These are treated as global defaults and apply to all the aggregate routes you configure in the `aggregate` statement.
- `route`—Here you configure individual aggregate routes. In this part of the `aggregate` statement, you optionally can configure aggregate route options. These options apply to the individual destination only and override any options you configured in the `defaults` part of the `aggregate` statement.

When you configure an individual aggregate route in the `route` part of the `aggregate` statement, specify the destination of the route (in `route` *`destination-prefix`*) in one of the following ways:

- *`network/mask-length`*, where *`network`* is the network portion of the IP address and *`mask-length`* is the destination prefix length.
- `default` if this is the default route to the destination. This is equivalent to specifying an IP address of `0.0.0.0/0`.

After you have configured aggregate routes, you can have a protocol advertise the routes by configuring a policy that is then exported by a routing protocol.

You can associate a routing policy when configuring an aggregate route's destination prefix in the `routes` part of the `aggregate` statement. Doing so provides the equivalent of an import routing policy filter for the destination prefix. That is, each potential contributor to an aggregate route, along with any aggregate options, is passed through the policy filter. The policy then can accept or reject the route as a contributor to the aggregate route and, if the contributor is accepted, the policy can modify the default preferences.

The following algorithm is used to compare two aggregate contributing routes in order to determine which one is the primary or preferred contributor:

1. Compare the protocol's `preferences` of the contributing routes. The lower the preference, the better the route. This is similar to the comparison that is done while determining the best route for the routing table.

2. Compare the protocol's `preferences2` of the contributing routes. The lower preference2 value is better. If only one route has `preferences2`, then this route is preferred.
3. The preference values are the same. Proceed with a numerical comparison of the prefix values.
   a. The primary contributor is the numerically smallest prefix value.
   b. If the two prefixes are numerically equal, the primary contributor is the route that has the smallest prefix length value.
4. At this point, the two routes are the same. The primary contributor does not change. An additional next hop is available for the existing primary contributor.

A rejected contributor still can contribute to a less specific aggregate route. If you do not specify a policy filter, all candidate routes contribute to an aggregate route.

To associate a routing policy with an aggregate route, include the `policy` statement when configuring the route:

```
aggregate (defaults | route) {
    policy policy-name;
}
```

In the `defaults` and `route` parts of the `aggregate` statement, you can specify *`aggregate-options`*, which define additional information about aggregate routes that is included with the route when it is installed in the routing table. All aggregate options are optional. Aggregate options that you specify in the `defaults` part of the `aggregate` statement are treated as global defaults and apply to all the aggregate routes you configure in the `aggregate` statement. Aggregate options that you specify in the `route` part of the `aggregate` statement override any global aggregate options and apply to that destination only.

To configure aggregate route options, include one or more of them in the `defaults` or `route` part of the `aggregate` statement:

```
[edit]
routing-options {
    aggregate {
        (defaults | route) {
            (active | passive);
            as-path <as-path> <origin (egp | igp | incomplete)> <atomic-aggregate> <aggregator as-
number in-address>;
            community [ community-ids ];
            discard;
            (brief | full);
            (metric | metric2 | metric3 | metric4) metric <type type>;
            (preference | preference2 | color | color2) preference <type type>;
            tag metric type number;
        }
    }
}
```

Configuring a Metric Value for Aggregate Routes

You can specify up to four metric values, starting with `metric` (for the first metric value) and continuing with `metric2`, `metric3`, and `metric4` by including one or more of the following statements:

```
aggregate (defaults | route) {
    (metric | metric2 | metric3 | metric4) metric <type type>;
}
```

For a list of hierarchy levels at which you can include these statements, see the statement summary sections for these statements.

In the `type` option, you can specify the type of route.

Configuring a Preference Value for Aggregate Routes

By default, aggregate routes have a preference value of 130. If the routing table contains a dynamic route to a destination that has a better (lower) preference value than this, the dynamic route is chosen as the active route and is installed in the forwarding table.

To modify the default preference value, specify a primary preference value (`preference`). You also can specify secondary preference value (`preference2`); and colors, which are even finer-grained preference values (`color` and `color2`). To do this, include one or more of the following statements:

```
aggregate (defaults | route) {
    (preference | preference2 | color | color2) preference <type type>;
}
```

For a list of hierarchy levels at which you can include these statements, see the statement summary sections for these statements.

The preference value can be a number in the range from 0 through 4,294,967,295 (2^{32} – 1) with a lower number indicating a more preferred route. For more information about preference values, see *Route Preferences Overview*.

In the `type` option, you can specify the type of route.

Configuring the Next Hop for Aggregate Routes

By default, when aggregate routes are installed in the routing table, the next hop is configured as a reject route. That is, the packet is rejected and an ICMP unreachable message is sent to the packet's originator.

When you configure an individual route in the `route` part of the `aggregate` statement, or when you configure the defaults for aggregate routes, you can specify a discard next hop. This means that if a more specific packet does not match a more specific route, the packet is rejected and a reject route for this destination is installed in the routing table, but ICMP unreachable messages are not sent.

Being able to discard next hops allows you to originate a summary route, which can be advertised through dynamic routing protocols, and allows you to discard received traffic that does not match a more specific route than the summary route. To discard next hops, include the `discard` option:

```
discard;
```

For a list of hierarchy levels at which you can include this statement, see the statement summary section for this statement.

Associating BGP Communities with Aggregate Routes

By default, no BGP community information is associated with aggregate routes. To associate community information with the routes, include the `community` option:

```
aggregate (defaults | route) {
    community [ community-ids ];
}
```

For a list of hierarchy levels at which you can include this statement, see the statement summary section for this statement. *`community-value`* is the community identifier and can be a number in the range from 0 through 65,535.

`community-ids` is one or more community identifiers for either communities or extended communities.

The format for community identifiers is:

```
as-number:community-value
```

as-number is the AS number and can be a value in the range from 1 through 65,534.

You also can specify *community-ids* for communities as one of the following well-known community names, which are defined in RFC 1997:

- `no-export`—Routes containing this community name are not advertised outside a BGP confederation boundary.
- `no-advertise`—Routes containing this community name are not advertised to other BGP peers.
- `no-export-subconfed`—Routes containing this community name are not advertised to external BGP peers, including peers in other members' ASs inside a BGP confederation.

You can explicitly exclude BGP community information with an aggregate route using the `none` option. Include `none` when configuring an individual route in the `route` portion of the `aggregate` statement to override a `community` option specified in the `defaults` portion of the statement.

NOTE: Extended community attributes are not supported at the `[edit routing-options]` hierarchy level. You must configure extended communities at the `[edit policy-options]` hierarchy level. For information about configuring extended communities information, see the "Configuring the Extended Communities Attribute" section in the Routing Policies, Firewall Filters, and Traffic Policers User Guide. For information about configuring 4-byte AS numbers and extended communities, see *Using 4-Byte Autonomous System Numbers in BGP Networks*.

Associating AS Paths with Aggregate Routes

By default, the AS path for aggregate routes is built from the component routes. To manually specify the AS path and associate AS path information with the routes, include the `as-path` option:

```
aggregate (defaults | route) {
    as-path <as-path> <origin (egp | igp | incomplete)> <atomic-aggregate> <aggregator as-number
in-address>;
}
```

For a list of hierarchy levels at which you can include this statement, see the statement summary section for this statement.

as-path is the AS path to include with the route. It can include a combination of individual AS path numbers and AS sets. Enclose sets in brackets ({ }). The first AS number in the path represents the AS immediately adjacent to the local AS. Each subsequent number represents an AS that is progressively farther from the local AS, heading toward the origin of the path.

NOTE: In Junos OS Release 9.1 and later, the numeric AS range is extended to provide BGP support for 4-byte AS numbers, as defined in RFC 4893, *BGP Support for Four-octet AS Number Space*. For the AS number, you can configure a value from 1 through 4,294,967,295. All releases of Junos OS support 2-byte AS numbers. The 2-byte AS number range is 1 through 65,535 (this is a subset of the 4-byte range).

In Junos OS Release 9.2 and later, you can also configure a 4-byte AS number using the AS-dot notation format of two integer values joined by a period: <*16-bit high-order value in decimal*>.<*16-bit low-order value in decimal*>. For example, the 4-byte AS number of 65,546 in plain-number format is represented as 1.10 in the AS-dot notation format. You can specify a value in the range from 0.0 through 65535.65535 in AS-dot notation format.

You also can specify the AS path using the BGP origin attribute, which indicates the origin of the AS path information:

- `egp`—Path information originated in another AS.
- `igp`—Path information originated within the local AS.
- `incomplete`—Path information was learned by some other means.

To attach the BGP `ATOMIC_AGGREGATE` path attribute to the aggregate route, specify the `atomic-aggregate` option. This path attribute indicates that the local system selected a less specific route rather than a more specific route.

To attach the BGP `AGGREGATOR` path attribute to the aggregate route, specify the `aggregator` option. When using this option, you must specify the last AS number that formed the aggregate route (encoded as two octets), followed by the IP address of the BGP system that formed the aggregate route.

NOTE: Starting with Junos OS 13.2R1, a BGP route is hidden when the AS path of an aggregate route—built from contributing routes— is more than half of the maximum BGP packet size (4096 bytes). Such AS paths have the OverflowASPathSize flag set for them. If you would like to leak such a BGP route, whose AS path length can overflow, we recommend to add the AS path statically in the default route configuration. For example:

```
[edit routing-instances instance-name routing options]
user@host# set aggregate route 0.0.0.0/0 as-path path 1267
```

Including AS Numbers in Aggregate Route Paths

By default, all AS numbers from all contributing paths are included in the aggregate route's path. To include only the longest common leading sequences from the contributing AS paths, include the `brief` option when configuring the route. If doing this results in AS numbers being omitted from the aggregate route, the BGP `ATOMIC_ATTRIBUTE` path attribute is included with the aggregate route.

```
aggregate (defaults | route) {
    brief;
}
```

To explicitly have all AS numbers from all contributing paths be included in the aggregate route's path, include the `full` option when configuring routes. Include this option when configuring an individual route in the `route` portion of the `aggregate` statement to override a `retain` option specified in the `defaults` portion of the statement.

```
aggregate (defaults | route) {
    full;
}
```

For a list of hierarchy levels at which you can include these statements, see the statement summary sections for these statements.

Configuring a Tag Value for Aggregate Routes

By default, no tag values are associated with aggregate routes. You can specify a tag value by including the `tag` option:

```
aggregate (defaults | route) {
    tag metric type number;
}
```

For a list of hierarchy levels at which you can include this statement, see the statement summary section for this statement.

Controlling Retention of Inactive Aggregate Routes in the Routing and Forwarding Tables

Static routes are only removed from the routing table if the next hop becomes unreachable, which happens if there are no contributing routes. To have an aggregate route remain continually installed in the routing and forwarding tables, include the `passive` option when configuring the route:

```
aggregate (defaults | route) {
    passive;
}
```

Routes that have been configured to remain continually installed in the routing and forwarding tables are marked with `reject` next hops when they are inactive.

To explicitly remove aggregate routes when they become inactive, include the `active` option when configuring routes. Include this option when configuring an individual route in the `route` portion of the `aggregate` statement to override a `passive` option specified in the `defaults` portion of the statement.

```
aggregate (defaults | route) {
    active;
}
```

Example: Summarizing Static Routes Through Route Aggregation

IN THIS SECTION

This example shows how to summarize routes by configuring aggregate routes.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

In this example:

- Device R1 is connected to customer networks 10.200.1.0/24 and 10.200.2.0/24.

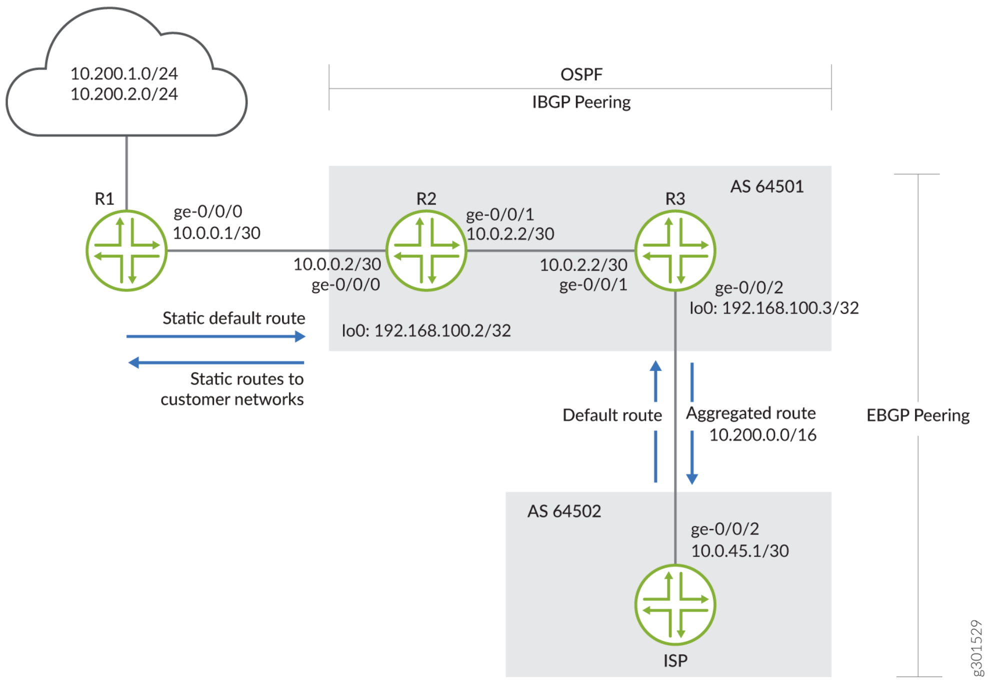
  For demonstration purposes, these routes are represented in this example as loopback interfaces on Device R1.
- Device R1 has a static default route to reach the ISP network (10.0.45.0).
- Device R2 has static routes configured to reach Device R1's customer networks (10.200.1.0/24 and 10.200.2.0/24).
- Device R2 also has a routing policy configured to advertise all static routes to its neighbor, Device R3.
- When Device R3 sends information about these routes (10.200.1.0/24 and 10.200.2.0/24) to Device ISP, the information is summarized as a single aggregate route (10.200.0.0/16).
- Device R2 and Device R3 share an IBGP session and have OSPF as the IGP.
- Device ISP injects a default route into AS 64501.

This example shows the configuration for all of the devices and the step-by-step configuration on Device R3.

Topology

Figure 10 on page 116 shows the sample network.

Figure 10: Aggregate Route Advertised to an ISP



Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

Device R1

```
set interfaces ge-0/0/0 description R1-to-R2
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 10.200.1.1/24
set interfaces lo0 unit 0 family inet address 10.200.2.2/24
set routing-options static route 0.0.0.0/0 next-hop 10.0.0.2
```

Device R2

```
set interfaces ge-0/0/0 description R2-to-R1
set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.2/30
set interfaces ge-0/0/1 description R2-to-R3
set interfaces ge-0/0/1 unit 0 family inet address 10.0.2.2/30
set interfaces lo0 unit 0 family inet address 192.168.100.2/32
set policy-options policy-statement send-customer-routes from protocol static
set policy-options policy-statement send-customer-routes then accept
set protocols bgp group internal type internal
set protocols bgp group internal local-address 192.168.100.2
set protocols bgp group internal export send-customer-routes
set protocols bgp group internal neighbor 192.168.100.3
set protocols ospf area 0.0.0.0 interface ge-0/0/0.0
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options autonomous-system 64501
set routing-options static route 10.200.1.0/24 next-hop 10.0.0.1
set routing-options static route 10.200.2.0/24 next-hop 10.0.0.1
```

Device R3

```
set interfaces ge-0/0/1 description R3-to-R2
set interfaces ge-0/0/1 unit 0 family inet address 10.0.2.1/30
set interfaces ge-0/0/2 description R3-to-ISP
set interfaces ge-0/0/2 unit 0 family inet address 10.0.45.2/30
set interfaces lo0 unit 0 family inet address 192.168.100.3/32
set policy-options policy-statement next-hop-self term 1 from protocol bgp
set policy-options policy-statement next-hop-self term 1 then next-hop self
set policy-options policy-statement next-hop-self term 1 then accept
set policy-options policy-statement send-aggregate term 1 from protocol aggregate
set policy-options policy-statement send-aggregate term 1 then accept
set policy-options policy-statement send-aggregate term suppress-specific-routes from route-
filter 10.200.0.0/16 longer
set policy-options policy-statement send-aggregate term suppress-specific-routes then reject
set protocols bgp group external type external
set protocols bgp group external export send-aggregate
set protocols bgp group external peer-as 64502
set protocols bgp group external neighbor 10.0.45.1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 192.168.100.3
set protocols bgp group internal export next-hop-self
set protocols bgp group internal neighbor 192.168.100.2
set protocols ospf area 0.0.0.0 interface ge-0/0/1.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options autonomous-system 64501
set routing-options aggregate route 10.200.0.0/16
```

Device ISP

```
set interfaces ge-0/0/2 description ISP-to-R3
set interfaces ge-0/0/2 unit 0 family inet address 10.0.45.1/30
set policy-options policy-statement advertise-default term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement advertise-default term 1 then accept
set protocols bgp group external type external
set protocols bgp group external export advertise-default
set protocols bgp group external peer-as 64501
set protocols bgp group external neighbor 10.0.45.2
set routing-options autonomous-system 64502
set routing-options static route 0.0.0.0/0 discard
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see Using the CLI Editor in Configuration Mode in the Junos OS CLI User Guide.

To configure Device R3:

1. Configure the device interfaces.

```
[edit interfaces]
user@R3# set ge-0/0/1 description R3-to-R2
user@R3# set ge-0/0/1 unit 0 family inet address 10.0.2.1/30

user@R3# set ge-0/0/2 description R3-to-ISP
user@R3# set  ge-0/0/2 unit 0 family inet address 10.0.45.2/30
```

```
user@R3# set lo0 unit 0 family inet address 192.168.100.3/32
```

2. Configure the AS number.

```
[edit routing-options]
user@R3# set autonomous-system 64501
```

3. Configure an EBGP session with the ISP device.

```
[edit protocols]
user@R3# set bgp group external type external
user@R3# set bgp group external peer-as 64502
user@R3# bgp group external neighbor 10.0.45.1
```

4. Configure an IBGP session with Device R2.

```
[edit protocols]
user@R3# set bgp group internal type internal
user@R3# set bgp group internal local-address 192.168.100.3
user@R3# set bgp group internal neighbor 192.168.100.2
```

5. Configure OSPF as the IGP.

```
[edit protocols]
user@R3# set ospf area 0.0.0.0 interface ge-0/0/1.0
user@R3# set ospf area 0.0.0.0 interface lo0.0 passive
```

6. Configure the aggregate route for the customer network routes.

```
[edit routing-options]
user@R3# set aggregate route 10.200.0.0/16
```

7. Configure a routing policy to advertise the aggregate route.

The first term in this policy advertises the aggregate route. The second term prevents more specific routes from being advertised.

```
[edit policy-options]
user@R3# set policy-statement send-aggregate term 1 from protocol aggregate
user@R3# set policy-statement send-aggregate term 1 then accept

user@R3# set policy-statement send-aggregate term suppress-specific-routes from route-
filter 10.200.0.0/16 longer
user@R3# set policy-statement send-aggregate term suppress-specific-routes then reject
```

8. Configure a routing policy to report Device R3 as the next hop as a result of participating in the EBGP session with Device ISP.

```
[edit policy-options]
user@R3# set policy-statement next-hop-self term 1 from protocol bgp
user@R3# set policy-statement next-hop-self term 1 then next-hop self
user@R3# set policy-statement next-hop-self term 1 then accept
```

9. Apply the aggregate route policy to the EBGP session with Device ISP.

```
[edit protocols]
user@R3# set bgp group external export send-aggregate
```

10. Apply the next-hop-self policy to the IBGP session with Device R2.

```
[edit protocols]
user@R3# set bgp group internal export next-hop-self
```

11. If you are done configuring the device, commit the configuration.

```
[edit]
user@R3# commit
```

Results

Confirm your configuration by issuing the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R3# show interfaces
ge-0/0/1 {
    description R3-to-R2;
    unit 0 {
        family inet {
            address 10.0.2.1/30;
        }
    }
}
ge-0/0/2 {
    description R3-to-ISP;
    unit 0 {
        family inet {
            address 10.0.45.2/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.100.3/32;
        }
    }
}
user@R3# show protocols
bgp {
    group external {
        type external;
        export send-aggregate;
        peer-as 64502;
        neighbor 10.0.45.1;
    }
    group internal {
        type internal;
        local-address 192.168.100.3;
        export next-hop-self;
        neighbor 192.168.100.2;
    }
}
ospf {
    area 0.0.0.0 {
        interface ge-0/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
user@R3# **show policy-options**
policy-statement next-hop-self {
    term 1 {
        from protocol bgp;
        then {
            next-hop self;
            accept;
        }
    }
}
policy-statement send-aggregate {
    term 1 {
        from protocol aggregate;
        then accept;
    }
    term suppress-specific-routes {
        from {
            route-filter 10.200.0.0/16 longer;
        }
        then reject;
    }
}
user@R3# **show routing-options**
autonomous-system 64501;
aggregate {
    route 10.200.0.0/16;
}
```

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying That Device R3 Has the Expected Routes

Purpose

Confirm that Device R3 has the advertised static routes from Device R2.

Action

```
user@R3>show route terse protocol bgp

inet.0: 12 destinations, 12 routes (12 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

A Destination        P Prf   Metric 1   Metric 2  Next hop        AS path
* 0.0.0.0/0          B 170        100            >10.0.45.1       64502 I
* 10.200.1.0/24      B 170        100            >10.0.2.2        I
* 10.200.2.0/24      B 170        100            >10.0.2.2        I
```

Meaning

The output shows that Device R3 has learned the static routes configured on Device R2 to reach Device R1's customer networks (10.200.1.0/24 and 10.200.2.0/24) through IBGP peering.

Verifying That Device R3 Advertises the Aggregate Route to Device ISP

Purpose

Make sure that Device R3 does not send the specific static routes and only sends the summarized aggregate route.

Action

```
user@R3>show route advertising-protocol bgp 10.0.45.1
inet.0: 20 destinations, 20 routes (20 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.200.0.0/16           Self                                    I
```

Meaning

The output shows that Device R3 sends only the summarized route to Device ISP.

Verifying End-to-End Connection

Purpose

Confirm end-to-end connection from the customer network on Device R1 to Device ISP.

Action

```
user@R1>ping 10.0.45.2 source 10.200.1.1
PING 10.0.45.2 (10.0.45.2): 56 data bytes
64 bytes from 10.0.45.2: icmp_seq=0 ttl=63 time=3.953 ms
64 bytes from 10.0.45.2: icmp_seq=1 ttl=63 time=4.979 ms
64 bytes from 10.0.45.2: icmp_seq=2 ttl=63 time=3.789 ms
```

Meaning

The output shows a successful ping verifying the reachability to Device ISP from customer network 10.200.1.1.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
13.2R1	Starting with Junos OS 13.2R1, a BGP route is hidden when the AS path of an aggregate route—built from contributing routes— is more than half of the maximum BGP packet size (4096 bytes).

RELATED DOCUMENTATION

Example: Configuring a Conditional Default Route Policy

5

CHAPTER

Generated Routes

IN THIS CHAPTER

Understanding Conditionally Generated Routes

Generated routes are used as the *route of last resort*. A packet is forwarded to the route of last resort when the routing tables have no information about how to reach that packet's destination. One use of route generation is to generate a default route to use if the routing table contains a route from a peer on a neighboring backbone.

A generated route becomes active when it has one or more *contributing routes*. A contributing route is an active route that is a more specific match for the generated destination. For example, for the destination 128.100.0.0/16, routes to 128.100.192.0/19 and 128.100.67.0/24 are contributing routes, but routes to 128.0.0.0./8, 128.0.0.0/16, and 128.100.0.0/16 are not.

A route can contribute only to a single generated route. However, an active generated route can recursively contribute to a less specific matching generated route. For example, a generated route to the destination 128.100.0.0/16 can contribute to a generated route to 128.96.0.0/13.

By default, when generated routes are installed in the routing table, the next hop is chosen from the primary contributing route.

NOTE: You can configure only one generated route for each destination prefix.

To configure generated routes in the default routing table (`inet.0`), include the `generate` statement:

```
generate {
    (defaults | route destination ) {
        (active | passive);
        as-path <as-path> <origin (egp | igp | incomplete)> <atomic-aggregate> <aggregator as-
number in-address>;
        community [ community-ids ];
        discard;
        (brief | full);
        (metric | metric2 | metric3 | metric4) metric <type type>;
        policy policy-name;
        (preference | preference2 | color | color2) preference <type type>;
        tag metric type number;
    }
}
```

NOTE: You cannot configure generated routes for the IPv4 multicast routing table (inet.1) or the IPv6 multicast routing table (`inet6.1`).

Starting from Junos OS release 15.1R3 onwards, generate route with table next-hop is supported.

The generate statement consists of two parts:

- `defaults`—Here you specify global generated route options. These are treated as global defaults and apply to all the generated routes you configure in the generate statement. This part of the generate statement is optional.
- `route`—Here you configure individual generated routes. In this part of the generate statement, you optionally can configure generated route options. These options apply to the individual destination only and override any options you configured in the defaults part of the generate statement.

RELATED DOCUMENTATION

Example: Configuring a Conditional Default Route Policy

Example: Configuring Filter-Based Forwarding on Logical Systems

6

CHAPTER

Martian Addresses

IN THIS CHAPTER

Recognize Martian Addresses for Routing

IN THIS SECTION

Understanding Martian Addresses

Martian addresses are host or network addresses about which all routing information is ignored. When received by the routing device, these routes are ignored. They commonly are sent by improperly configured systems on the network and have destination addresses that are obviously invalid.

In IPv6, the loopback address and the multicast resolve and discard routes are the default martian addresses.

In Junos OS Release 10.4R5 and later, the reserved IPv6 multicast address space (ff00::/8 and ff02::/16) is added to the list of martian addresses.

In Junos OS Release 9.6 and later, you can configure Class E addresses on interfaces. Class E addresses are treated like any other unicast address for the purpose of forwarding. To allow Class E addresses to be configured on interfaces, you must remove the Class E prefix from the list of martian addresses. To remove the Class E prefix from the list of martian addresses include the `martians 240/4 orlonger allow` statement at the `[edit routing-options]` hierarchy level.

To view the default and configured martian routes, run the `show route martians` command.

IPv4 Martian Addresses

```
user@host> show route martians table inet.

inet.0:
             0.0.0.0/0 exact -- allowed
             0.0.0.0/8 orlonger -- disallowed
             127.0.0.0/8 orlonger -- disallowed
             192.0.0.0/24 orlonger -- disallowed
             240.0.0.0/4 orlonger -- disallowed
             224.0.0.0/4 exact -- disallowed
             224.0.0.0/24 exact -- disallowed

inet.1:
             0.0.0.0/0 exact -- allowed
             0.0.0.0/8 orlonger -- disallowed
             127.0.0.0/8 orlonger -- disallowed
             192.0.0.0/24 orlonger -- disallowed
             240.0.0.0/4 orlonger -- disallowed

inet.2:
             0.0.0.0/0 exact -- allowed
             0.0.0.0/8 orlonger -- disallowed
             127.0.0.0/8 orlonger -- disallowed
             192.0.0.0/24 orlonger -- disallowed
             240.0.0.0/4 orlonger -- disallowed
             224.0.0.0/4 exact -- disallowed
             224.0.0.0/24 exact -- disallowed

inet.3:
             0.0.0.0/0 exact -- allowed
             0.0.0.0/8 orlonger -- disallowed
             127.0.0.0/8 orlonger -- disallowed
             192.0.0.0/24 orlonger -- disallowed
             240.0.0.0/4 orlonger -- disallowed
             224.0.0.0/4 exact -- disallowed
             224.0.0.0/24 exact -- disallowed
```

IPv6 Martian Addresses

```
user@host> show route martians table inet6
inet6.0:
             ::1/128 exact -- disallowed
             ff00::/8 exact -- disallowed
             ff02::/16 exact -- disallowed

inet6.1:
             ::1/128 exact -- disallowed

inet6.2:
             ::1/128 exact -- disallowed
             ff00::/8 exact -- disallowed
             ff02::/16 exact -- disallowed
```

```
inet6.3:
             ::1/128 exact -- disallowed
             ff00::/8 exact -- disallowed
             ff02::/16 exact -- disallowed
```

Example: Removing the Class E Prefix on Martian Addresses

IN THIS SECTION

This example shows how to remove the Class E prefix from the list of martian addresses.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, Junos OS defaults are modified to allow the 240.0.0.0/4 address block. This block of addresses is known as the experimental Class E addresses. In Junos OS Release 9.6 and later, you can configure Class E addresses on interfaces and use them for forwarding traffic. However, to do this, you must first allow routing on this address block.

This example also shows how to modify the martian addresses in the IPv6 routing table, **inet6.0**.

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

```
set routing-options rib inet.1 martians 240.0.0.0/4 orlonger allow
set routing-options rib inet6.0 martians fd00::/8 orlonger
set routing-options rib inet.3 martians 240.0.0.0/4 orlonger allow
set routing-options rib inet.2 martians 240.0.0.0/4 orlonger allow
set routing-options martians 240.0.0.0/4 orlonger allow
```

Procedure

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure martian routes:

1. Allow Class E addresses in the default unicast routing table.

   ```
   [edit routing-options]
   user@host# set martians 240.0.0.0/4 orlonger allow
   ```

2. Allow Class E addresses in the routing table that is used for the IPv4 multicast forwarding cache.

   ```
   [edit routing-options]
   user@host# set rib inet.1 martians 240.0.0.0/4 orlonger allow
   ```

3. Allow Class E addresses in the routing table that is used for multicast reverse path forwarding (RPF) lookup.

```
[edit routing-options]
user@host# set rib inet.2 martians 240.0.0.0/4 orlonger allow
```

4. Allow Class E addresses in the routing table that stores MPLS LSP information.

```
[edit routing-options]
user@host# set rib inet.3 martians 240.0.0.0/4 orlonger allow
```

5. Add a disallowed martian route to the IPv6 unicast routing table.

```
[edit routing-options]
user@host# set rib inet6.0 martians fd00::/8 orlonger
```

6. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Results

Confirm your configuration by issuing the `show routing-options` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show routing-options
rib inet.1 {
    martians {
        240.0.0.0/4 orlonger allow;
    }
}
rib inet6.0 {
    martians {
        fd00::/8 orlonger;
    }
}
rib inet.3 {
    martians {
        240.0.0.0/4 orlonger allow;
    }
}
rib inet.2 {
    martians {
        240.0.0.0/4 orlonger allow;
    }
}
martians {
    240.0.0.0/4 orlonger allow;
}
```

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying That the 240.0.0.0/4 Routes Are Now Accepted

Purpose

Make sure that the 240.0.0.0/4 route appears in the routing tables as allowed.

Action

```
user@host> show route martians table inet.
inet.0:
             0.0.0.0/0 exact -- allowed
             0.0.0.0/8 orlonger -- disallowed
             127.0.0.0/8 orlonger -- disallowed
             192.0.0.0/24 orlonger -- disallowed
             240.0.0.0/4 orlonger -- allowed
             224.0.0.0/4 exact -- disallowed
```

```
            224.0.0.0/24 exact -- disallowed

inet.1:
             0.0.0.0/0 exact -- allowed
             0.0.0.0/8 orlonger -- disallowed
             127.0.0.0/8 orlonger -- disallowed
             192.0.0.0/24 orlonger -- disallowed
             240.0.0.0/4 orlonger -- allowed

inet.2:
             0.0.0.0/0 exact -- allowed
             0.0.0.0/8 orlonger -- disallowed
             127.0.0.0/8 orlonger -- disallowed
             192.0.0.0/24 orlonger -- disallowed
             240.0.0.0/4 orlonger -- allowed
             224.0.0.0/4 exact -- disallowed
             224.0.0.0/24 exact -- disallowed

inet.3:
             0.0.0.0/0 exact -- allowed
             0.0.0.0/8 orlonger -- disallowed
             127.0.0.0/8 orlonger -- disallowed
             192.0.0.0/24 orlonger -- disallowed
             240.0.0.0/4 orlonger -- allowed
             224.0.0.0/4 exact -- disallowed
             224.0.0.0/24 exact -- disallowed
```

Meaning

The output shows that the 240.0.0.0/4 route is allowed.

Verifying That the fd00::/8 Routes Are Now Rejected

Purpose

Make sure that the fd00::/8 route appears in the IPv6 unicast routing table as disallowed.

Action

```
user@host> show route martians table inet6.0
inet6.0:
```

```
               ::1/128 exact -- disallowed
               ff00::/8 exact -- disallowed
               ff02::/16 exact -- disallowed
               fd00::/8 orlonger -- disallowed
```

Meaning

The output shows that the fd00::/8 route is disallowed.

RELATED DOCUMENTATION

Example: Creating an Interface on a Logical System

Example: Configuring an OSPF Default Route Policy on Logical Systems

7

CHAPTER

Packet Forwarding

IN THIS CHAPTER

Configuring Packet Forwarding Behavior

IN THIS SECTION

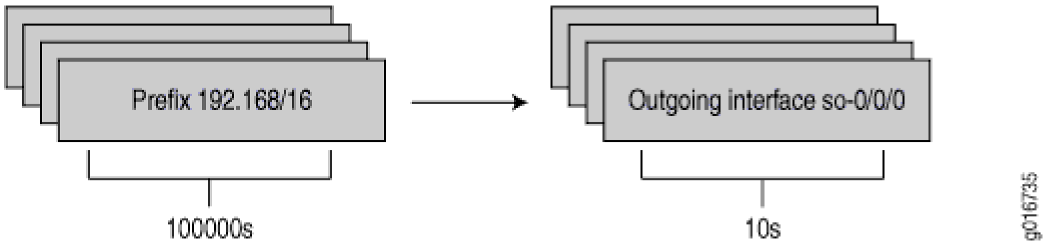
Understanding Indirect Next Hops

Junos OS supports the concept of an indirect next hop for all routing protocols that support indirectly connected next hops, also known as third-party next hops.

Because routing protocols such as internal BGP (IBGP) can send routing information about indirectly connected routes, Junos OS relies on routes from intra-AS routing protocols (OSPF, IS-IS, RIP, and static) to resolve the best directly connected next hop. The Routing Engine performs route resolution to determine the best directly connected next hop and installs the route to the Packet Forwarding Engine.

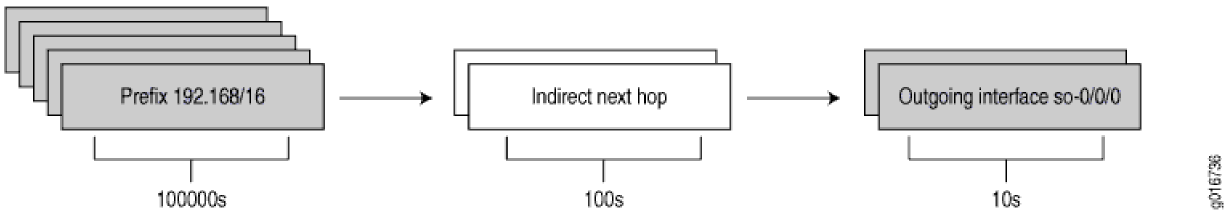
By default, Junos OS does not maintain the route for indirect next hop to forwarding next-hop binding on the Packet Forwarding Engine forwarding table. As a result, when a rerouting event occurs, potentially thousands of route to forwarding next-hop bindings must be updated, which increases the route convergence time. Figure 11 on page 139 illustrates the route to forwarding next-hop bindings with indirect next hop disabled.

Figure 11: Route to Forwarding Next-Hop Bindings



You can enable Junos OS to maintain the indirect next hop to forwarding next-hop binding on the Packet Forwarding Engine forwarding table. As a result, fewer route to forwarding next-hop bindings need to be updated, which improves the route convergence time. Figure 12 on page 140 illustrates the route to forwarding next-hop bindings with indirect next hop enabled.

Figure 12: Route to Forwarding Indirect Next-Hop Bindings



Example: Optimizing Route Reconvergence by Enabling Indirect Next Hops on the Packet Forwarding Engine

IN THIS SECTION

This example shows how to use indirect next hops to promote faster network convergence (for example, in BGP networks) by decreasing the number of forwarding table changes required when a change in the network topology occurs.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

IN THIS SECTION

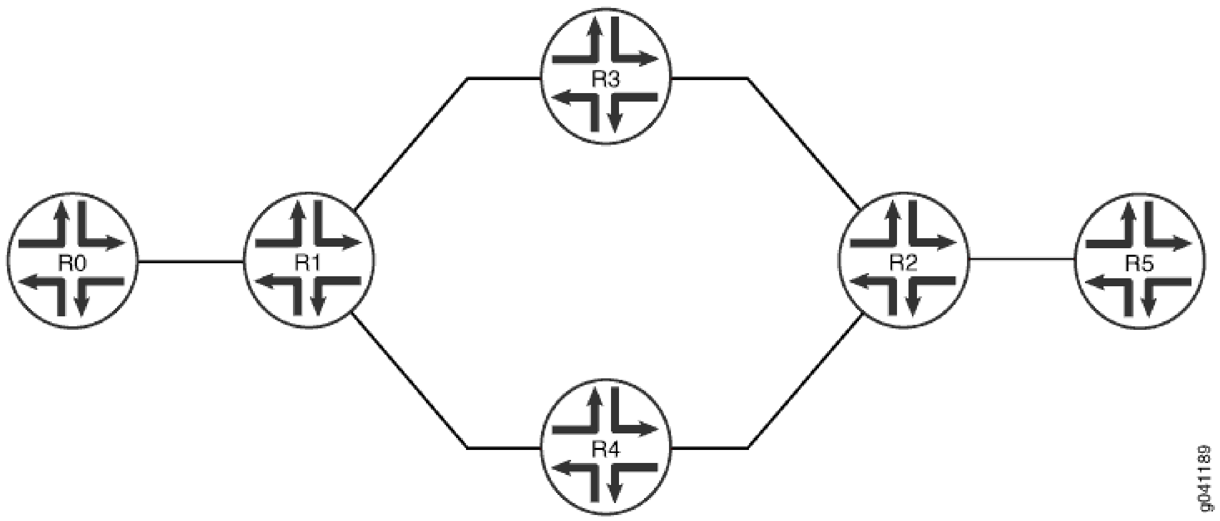
In this example, several devices are connected over unequal-cost paths. From Device R1 to Device R2, the path through Device R3 has a higher IGP metric than the path through Device R4. Device R1 has an internal BGP connection to Device R2. Device R0 injects multiple routes into the network, and Device R1 advertises those routes to Device R2. Because Device R2 is not directly connected to Device R1, Device R2's forwarding table contains indirect next hops. An interior gateway protocol, in this case OSPF, is running on the internal links among Devices R1, R2, R3, and R4. Each router is advertising its loopback interface IPv4 address.

On Device R2, the `indirect-next-hop` statement enables Junos OS to maintain the indirect next hop to forwarding next-hop binding on the Packet Forwarding Engine forwarding table. As a result, fewer route to forwarding next-hop bindings need to be updated, which improves the route convergence time if a path fails.

Topology

Figure 13 on page 141 shows the sample network.

Figure 13: Sample Topology for Indirect Next Hops



The "CLI Quick Configuration" on page 142 section shows the full configuration on all of the devices in Figure 13 on page 141. Otherwise, the example focuses on Device R0, Device R1, and Device R2.

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

Device R0

```
set interfaces fe-1/2/0 unit 1 family inet address 10.0.0.1/30
set interfaces lo0 unit 1 family inet address 10.1.0.1/32
set interfaces lo0 unit 1 family inet address 10.1.0.2/32
set interfaces lo0 unit 1 family inet address 10.1.0.3/32
set interfaces lo0 unit 1 family inet address 10.1.0.4/32
set interfaces lo0 unit 1 family inet address 10.1.0.5/32
set interfaces lo0 unit 1 family inet address 10.1.0.6/32
set interfaces lo0 unit 1 family inet address 10.1.0.7/32
set interfaces lo0 unit 1 family inet address 10.1.0.8/32
set interfaces lo0 unit 1 family inet address 10.1.0.9/32
set routing-options static route 0.0.0.0/0 next-hop 10.0.0.2
```

Device R1

```
set interfaces fe-1/2/0 unit 2 family inet address 10.0.0.2/30
set interfaces fe-1/2/1 unit 5 family inet address 10.0.0.5/30
set interfaces fe-1/2/2 unit 9 family inet address 10.0.0.9/30
set interfaces lo0 unit 2 family inet address 10.1.1.1/32
set protocols bgp export send-local
set protocols bgp export send-static
set protocols bgp group int type internal
set protocols bgp group int local-address 10.1.1.1
set protocols bgp group int neighbor 10.2.2.2
set protocols ospf area 0.0.0.0 interface fe-1/2/1.5
set protocols ospf area 0.0.0.0 interface fe-1/2/2.9
set protocols ospf area 0.0.0.0 interface lo0.2
set policy-options policy-statement send-local from protocol local
set policy-options policy-statement send-local from protocol direct
set policy-options policy-statement send-local then accept
set policy-options policy-statement send-static from protocol static
set policy-options policy-statement send-static then accept
set routing-options static route 10.1.0.2/32 next-hop 10.0.0.1
set routing-options static route 10.1.0.1/32 next-hop 10.0.0.1
set routing-options static route 10.1.0.3/32 next-hop 10.0.0.1
set routing-options static route 10.1.0.4/32 next-hop 10.0.0.1
set routing-options static route 10.1.0.5/32 next-hop 10.0.0.1
set routing-options static route 10.1.0.6/32 next-hop 10.0.0.1
set routing-options static route 10.1.0.7/32 next-hop 10.0.0.1
set routing-options static route 10.1.0.8/32 next-hop 10.0.0.1
set routing-options static route 10.1.0.9/32 next-hop 10.0.0.1
set routing-options autonomous-system 65500
```

Device R2

```
set interfaces fe-1/2/0 unit 14 family inet address 10.0.0.14/30
set interfaces fe-1/2/1 unit 18 family inet address 10.0.0.18/30
set interfaces fe-1/2/2 unit 21 family inet
set interfaces lo0 unit 3 family inet address 10.2.2.2/32
set protocols bgp export send-local
set protocols bgp group int type internal
set protocols bgp group int local-address 10.2.2.2
set protocols bgp group int family inet unicast
set protocols bgp group int family inet-vpn unicast
set protocols bgp group int neighbor 10.1.1.1
set protocols ospf area 0.0.0.0 interface fe-1/2/0.14
set protocols ospf area 0.0.0.0 interface fe-1/2/1.18
set protocols ospf area 0.0.0.0 interface lo0.3
set policy-options policy-statement send-local from protocol local
set policy-options policy-statement send-local from protocol direct
set policy-options policy-statement send-local then accept
set routing-options autonomous-system 65500
set routing-options forwarding-table indirect-next-hop
```

Device R3

```
set interfaces fe-1/2/0 unit 6 family inet address 10.0.0.6/30
set interfaces fe-1/2/1 unit 13 family inet address 10.0.0.13/30
set interfaces lo0 unit 4 family inet address 10.3.3.3/32
set protocols ospf area 0.0.0.0 interface fe-1/2/0.6 metric 5000
set protocols ospf area 0.0.0.0 interface fe-1/2/1.13 metric 5000
set protocols ospf area 0.0.0.0 interface lo0.4
```

Device R4

```
set interfaces fe-1/2/0 unit 10 family inet address 10.0.0.10/30
set interfaces fe-1/2/1 unit 17 family inet address 10.0.0.17/30
set interfaces lo0 unit 5 family inet address 10.4.4.4/32
set protocols ospf area 0.0.0.0 interface fe-1/2/0.10
set protocols ospf area 0.0.0.0 interface fe-1/2/1.17
set protocols ospf area 0.0.0.0 interface lo0.5
```

Device R5

```
set interfaces fe-1/2/0 unit 22 family inet address 10.0.0.22/30
set interfaces lo0 unit 6 family inet address 10.5.5.5/32
```

Configuring Device R0

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure Device R0:

1. Configure the interfaces, including multiple routes that can be injected into the network for demonstration purposes.

   ```
   [edit interfaces]
   user@R0# set fe-1/2/0 unit 1 family inet address 10.0.0.1/30
   user@R0# set lo0 unit 1 family inet address 10.1.0.1/32
   user@R0# set lo0 unit 1 family inet address 10.1.0.2/32
   ```

```
user@R0# set lo0 unit 1 family inet address 10.1.0.3/32
user@R0# set lo0 unit 1 family inet address 10.1.0.4/32
user@R0# set lo0 unit 1 family inet address 10.1.0.5/32
user@R0# set lo0 unit 1 family inet address 10.1.0.6/32
user@R0# set lo0 unit 1 family inet address 10.1.0.7/32
user@R0# set lo0 unit 1 family inet address 10.1.0.8/32
user@R0# set lo0 unit 1 family inet address 10.1.0.9/32
```

2. Configure a static default route for network reachability.

```
[edit routing-options]
user@R0# set static route 0.0.0.0/0 next-hop 10.0.0.2
```

3. If you are done configuring the device, commit the configuration.

```
[edit]
user@R0# commit
```

Configuring Device R1

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure Device R1:

1. Configure the interfaces, including multiple routes that can be injected into the network for demonstration purposes.

```
[edit interfaces]
user@R1# set fe-1/2/0 unit 2 family inet address 10.0.0.2/30
user@R1# set fe-1/2/1 unit 5 family inet address 10.0.0.5/30
user@R1# set fe-1/2/2 unit 9 family inet address 10.0.0.9/30
user@R1# set lo0 unit 2 family inet address 10.1.1.1/32
```

2. Configure BGP.

```
[edit protocols]
user@R1# set bgp export send-local
user@R1# set bgp export send-static
user@R1# set bgp group int type internal
user@R1# set bgp group int local-address 10.1.1.1
user@R1# set bgp group int neighbor 10.2.2.2
```

3. Configure OSPF.

```
[edit protocols]
user@R1# set ospf area 0.0.0.0 interface fe-1/2/1.5
user@R1# set ospf area 0.0.0.0 interface fe-1/2/2.9
user@R1# set ospf area 0.0.0.0 interface lo0.2
```

4. Configure the routing policies.

```
[edit]
user@R1# set policy-options policy-statement send-local from protocol local
user@R1# set policy-options policy-statement send-local from protocol direct
user@R1# set policy-options policy-statement send-local then accept
user@R1# set policy-options policy-statement send-static from protocol static
user@R1# set policy-options policy-statement send-static then accept
```

5. Configure a set of static routes to the set of interfaces configured on Device R0.

```
[edit]
user@R1# set routing-options static route 10.1.0.2/32 next-hop 10.0.0.1
user@R1# set routing-options static route 10.1.0.1/32 next-hop 10.0.0.1
user@R1# set routing-options static route 10.1.0.3/32 next-hop 10.0.0.1
user@R1# set routing-options static route 10.1.0.4/32 next-hop 10.0.0.1
user@R1# set routing-options static route 10.1.0.5/32 next-hop 10.0.0.1
user@R1# set routing-options static route 10.1.0.6/32 next-hop 10.0.0.1
user@R1# set routing-options static route 10.1.0.7/32 next-hop 10.0.0.1
user@R1# set routing-options static route 10.1.0.8/32 next-hop 10.0.0.1
user@R1# set routing-options static route 10.1.0.9/32 next-hop 10.0.0.1
```

6. Configure the autonomous system (AS) identifier.

```
[edit]
user@R1# set routing-options autonomous-system 65500
```

7. If you are done configuring the device, commit the configuration.

```
[edit]
user@R1# commit
```

Configuring Device R2

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure Device R2:

1. Configure the interfaces, including multiple routes that can be injected into the network for demonstration purposes.

```
[edit interfaces]
user@R2# set fe-1/2/0 unit 14 family inet address 10.0.0.14/30
user@R2# set fe-1/2/1 unit 18 family inet address 10.0.0.18/30
user@R2# set fe-1/2/2 unit 21 family inet address 10.0.0.21/30;
user@R2# set lo0 unit 3 family inet address 10.2.2.2/32
```

2. Configure BGP.

```
[edit]
user@R2# set protocols bgp export send-local
user@R2# set protocols bgp group int type internal
user@R2# set protocols bgp group int local-address 10.2.2.2
user@R2# set protocols bgp group int family inet unicast
user@R2# set protocols bgp group int family inet-vpn unicast
user@R2# set protocols bgp group int neighbor 10.1.1.1
```

3. Configure OSPF.

```
[edit]
user@R2# set protocols ospf area 0.0.0.0 interface fe-1/2/0.14
user@R2# set protocols ospf area 0.0.0.0 interface fe-1/2/1.18
user@R2# set protocols ospf area 0.0.0.0 interface lo0.3
```

4. Configure the routing policies.

```
[edit]
user@R2# set policy-options policy-statement send-local from protocol local
user@R2# set policy-options policy-statement send-local from protocol direct
user@R2# set policy-options policy-statement send-local then accept
```

5. Configure the AS identifier.

```
[edit]
user@R2# set routing-options autonomous-system 65500
```

6. Enable indirect next hops in the forwarding plane.

```
[edit]
user@R2# set routing-options forwarding-table indirect-next-hop
```

7. If you are done configuring the device, commit the configuration.

```
[edit]
user@R2# commit
```

Results

Confirm your configuration by issuing the `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device R0

```
user@R0# show interfaces
fe-1/2/0 {
    unit 1 {
        family inet {
            address 10.0.0.1/30;
        }
    }
}
lo0 {
    unit 1 {
        family inet {
            address 10.1.0.1/32;
            address 10.1.0.2/32;
            address 10.1.0.3/32;
            address 10.1.0.4/32;
            address 10.1.0.5/32;
            address 10.1.0.6/32;
            address 10.1.0.7/32;
            address 10.1.0.8/32;
            address 10.1.0.9/32;
        }
    }
}
```

```
user@R0# show routing-options
static {
    route 0.0.0.0/0 next-hop 10.0.0.2;
}
```

Device R1

```
user@R1# show interfaces
fe-1/2/0 {
    unit 2 {
        family inet {
            address 10.0.0.2/30;
        }
    }
}
fe-1/2/1 {
    unit 5 {
        family inet {
            address 10.0.0.5/30;
        }
    }
}
fe-1/2/2 {
    unit 9 {
        family inet {
            address 10.0.0.9/30;
        }
    }
}
lo0 {
    unit 2 {
        family inet {
            address 10.1.1.1/32;
        }
    }
}
```

```
user@R1# show protocols
bgp {
    export [ send-local send-static ];
    group int {
        type internal;
        local-address 10.1.1.1;
        neighbor 10.2.2.2;
    }
}
ospf {
    area 0.0.0.0 {
        interface fe-1/2/1.5;
        interface fe-1/2/2.9;
        interface lo0.2;
    }
}
```

```
user@R1# show policy-options
policy-statement send-local {
    from protocol [ local direct ];
    then accept;
}
policy-statement send-static {
    from protocol static;
    then accept;
}
```

```
user@R1# show routing-options
static {
    route 10.1.0.2/32 next-hop 10.0.0.1;
    route 10.1.0.1/32 next-hop 10.0.0.1;
    route 10.1.0.3/32 next-hop 10.0.0.1;
    route 10.1.0.4/32 next-hop 10.0.0.1;
    route 10.1.0.5/32 next-hop 10.0.0.1;
    route 10.1.0.6/32 next-hop 10.0.0.1;
    route 10.1.0.7/32 next-hop 10.0.0.1;
    route 10.1.0.8/32 next-hop 10.0.0.1;
    route 10.1.0.9/32 next-hop 10.0.0.1;
}
autonomous-system 65500;
```

Device R2

```
user@R2# show interfaces
fe-1/2/0 {
    unit 14 {
        family inet {
            address 10.0.0.14/30;
        }
    }
}
fe-1/2/1 {
    unit 18 {
        family inet {
            address 10.0.0.18/30;
        }
    }
}
fe-1/2/2 {
    unit 21 {
        family inet {
            address 10.0.0.21/30
        }
    }
}
lo0 {
    unit 3 {
        family inet {
            address 10.2.2.2/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    export send-local;
    group int {
        type internal;
        local-address 10.2.2.2;
        family inet {
            unicast;
        }
        family inet-vpn {
            unicast;
        }
        neighbor 10.1.1.1;
    }
}
ospf {
    area 0.0.0.0 {
        interface fe-1/2/0.14;
        interface fe-1/2/1.18;
        interface lo0.3;
    }
}
```

```
user@R2# show policy-options
policy-statement send-local {
    from protocol [ local direct ];
    then accept;
}
```

```
user@R2# show routing-options
autonomous-system 65500;
forwarding-table {
    indirect-next-hop;
}
```

Configure Device R3, Device R4, and Device R5, as shown in "CLI Quick Configuration" on page 142.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying That the Routes Have the Expected Indirect-Next-Hop Flag

Purpose

Make sure that Device R2 is configured to maintain the indirect next hop to forwarding next-hop binding on the Packet Forwarding Engine forwarding table.

Action

```
user@R2> show krt indirect-next-hop
show krt indirect-next-hop
```

```
Indirect Nexthop:
Index: 1048575 Protocol next-hop address: 10.255.3.1
  RIB Table: __mpls-oam__.mpls.0
  Label: Swap 299968
  Policy Version: 0                     References: 1
  Locks: 2                              0x95bc514
  Flags: 0x3
  INH Session ID: 0xa
  INH Version ID: 1
  Ref RIB Table: unknown
        Next hop: 10.50.244.9 via ge-2/0/2.0
        Label operation: Swap 299968, Push 299792(top)
        Label TTL action: no-prop-ttl, no-prop-ttl(top)
        Session Id: 0x9
      IGP FRR Interesting proto count : 0
```

Meaning

The `0x3` flag in the output indicates that Device R2 is configured to maintain the indirect next hop to forwarding next-hop binding on the Packet Forwarding Engine forwarding table. When the `indirect-next-hop` statement is deleted or deactivated from the configuration, this flag changes to `0x2`. Junos MX series routers with Trio Modular Port Concentrator (MPC) chipset supports indirect-next-hop by default and can not be disabled. Thus, even if `indirect-next-hop` is not configured under `forwarding-options`, the feature will work by default. Thus, `0x3` flag is not applicable for Trio Modular Port Concentrator (MPCs).

NOTE: The `show krt indirect-next-hop` command is hidden and is therefore undocumented. The `show krt indirect-next-hop` command is shown here because this is the only command that verifies the indirect next-hop feature. The best verification method is, of course, monitoring network performance during reconvergence after a path failure.

Packet Forwarding Engine Install Error Checking

SUMMARY

(Junos OS Evolved only) This error checking mechanism enhances network resiliency by enabling the forwarding plane to notify the control plane of any next-hop installation errors, thereby maintaining accurate routing states and preventing traffic loss. Errors from multiple Packet Forwarding Engines are aggregated and reported in a consolidated manner, simplifying error detection and resolution processes.

IN THIS SECTION

We have more than one copy of routing information present in a router. Two such repositories are the Routing Information Base (RIB) and the Forwarding Information Base (FIB). The control plane computes the data flow packet path and stores the data in the RIB. A subset of this data is propagated to the forwarding plane ASIC and is stored in the FIB, which finally gets programmed in the forwarding ASIC in the Packet Forwarding Engine (PFE).

Sometimes the routing information present in the RIB is not correctly installed in the FIB. The control plane owning the RIB might not have information about the discrepancy between the contents of the RIB and that in the forwarding plane. This discrepancy might result in the control plane protocol publishing a state of the routing path that is not accurate. Eventually this might lead to other problems, including traffic loss.

Benefits of Packet Forwarding Engine Install Error Checking

- Enables the forwarding plane to notify the control plane of any next-hop installation errors, thereby maintaining accurate routing states and preventing traffic loss.
- Errors from multiple Packet Forwarding Engines are aggregated and reported in a consolidated manner, simplifying error detection and resolution processes.
- Covers various next-hop types, ensuring broad error detection and notification, which contributes to the reliability of network operations.
-

Enable and Monitor PFE Install Error Checking

To enable this error checking, configure the `pfe-install-error` statement at the `[edit routing-options forwarding-table]` hierarchy level, then restart the rpd process. Two operational mode CLI commands are available to check on these route errors after you have configured this feature:

- If the forwarding plane is not forwarding traffic, use the `show route pfe-install-error` command to check if the Packet Forwarding Engine has sent an error for the route that is dropping traffic.

- To check for next-hop errors, use the `show routing nhdb pfe-install-error` command. We support errors for routes pointing to a limited set of next hops: RT_NH_INDIRECT, RT_NH_ROUTER, RT_NH_CHAIN, RT_NH_INDXD, RT_NH_LIST, RT_NH_TUNNEL_COMP, and RT_NH_FRR_INDIRECT .

This feature does support sharding, ensuring that routes with sharding enabled will also handle next-hop errors appropriately without breaking the sharding functionality. In scaled scenarios, not all errors generated may be displayed due to limitations in error message handling. The Packet Forwarding Engine will only report next-hop installation failures, which are the most probable type of error.

RELATED DOCUMENTATION

forwarding-table

8

CHAPTER

Troubleshooting

IN THIS CHAPTER

Troubleshooting Network Issues

IN THIS SECTION

Working with Problems on Your Network

IN THIS SECTION

Problem

Description

This checklist provides links to troubleshooting basics, an example network, and includes a summary of the commands you might use to diagnose problems with the router and network.

Solution

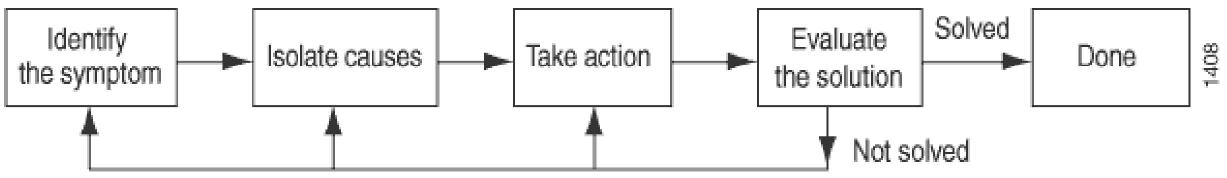
Table 2: Checklist for Working with Problems on Your Network

Tasks	Command or Action
Isolating a Broken Network Connection	
1. *Identifying the Symptoms of a Broken Network Connection*	**ping (*ip-address* | *hostname*) show route (*ip-address* | *hostname*) traceroute (*ip-address* | *hostname*)**
1. *Isolating the Causes of a Network Problem*	`show < configuration | interfaces | protocols | route >`
1. *Taking Appropriate Action for Resolving the Network Problem*	`[edit] delete routing options static route` *`destination-prefix`* **commit and-quit show route *destination-prefix***
1. *Evaluating the Solution to Check Whether the Network Problem Is Resolved*	`show route (`*`ip-address`*` | `*`hostname`*`) ping (`*`ip-address`*` | `*`hostname`*`)` **count 3 traceroute (*ip-address* | *hostname*)**

Isolating a Broken Network Connection

By applying the standard four-step process illustrated in Figure 14 on page 159, you can isolate a failed node in the network. Note that the functionality described in this section is not supported in versions 15.1X49, 15.1X49-D30, or 15.1X49-D40.

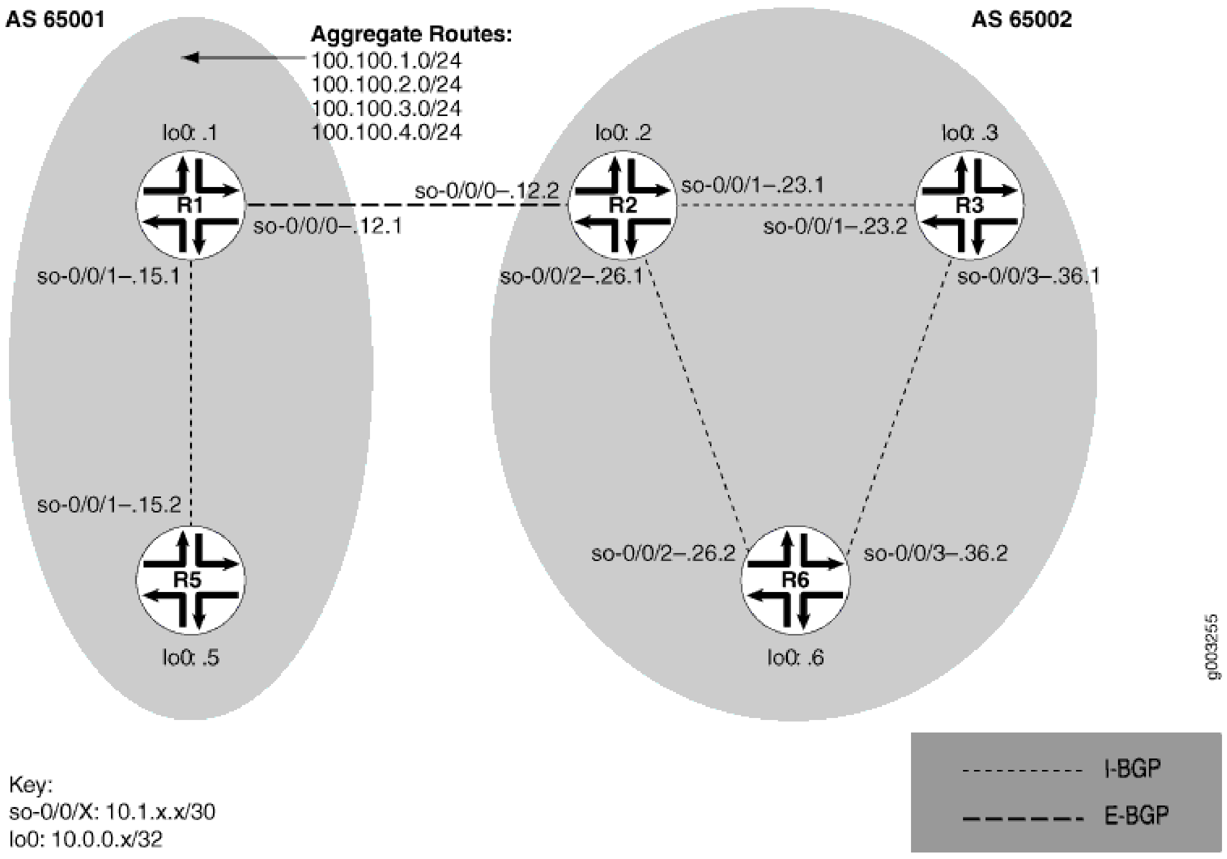
Figure 14: Process for Diagnosing Problems in Your Network



Before you embark on the four-step process, however, it is important that you are prepared for the inevitable problems that occur on all networks. While you might find a solution to a problem by simply trying a variety of actions, you can reach an appropriate solution more quickly if you are systematic in your approach to the maintenance and monitoring of your network. To prepare for problems on your network, understand how the network functions under normal conditions, have records of baseline network activity, and carefully observe the behavior of your network during a problem situation.

Figure 15 on page 160 shows the network topology used in this topic to illustrate the process of diagnosing problems in a network.

Figure 15: Network with a Problem



The network in Figure 15 on page 160 consists of two autonomous systems (ASs). AS 65001 includes two routers, and AS 65002 includes three routers. The border router (`R1`) in AS 65001 announces aggregated prefixes `100.100/24` to the AS 65002 network. The problem in this network is that `R6` does not have access to `R5` because of a loop between `R2` and `R6`.

To isolate a failed connection in your network, follow the steps in these topics:

- *Isolating the Causes of a Network Problem*

- *Taking Appropriate Action for Resolving the Network Problem*
- *Taking Appropriate Action for Resolving the Network Problem*
- *Evaluating the Solution to Check Whether the Network Problem Is Resolved*

Identifying the Symptoms of a Broken Network Connection

IN THIS SECTION

Problem

Description

The symptoms of a problem in your network are usually quite obvious, such as the failure to reach a remote host.

Solution

To identify the symptoms of a problem on your network, start at one end of your network and follow the routes to the other end, entering all or one of the following Junos OS command-line interfaces (CLI) operational mode commands:

```
           user@host> ping (ip-address | host-name)
user@host> show route (ip-address | host-name)
user@host> traceroute (ip-address | host-name)
```

Sample Output

```
user@R6> ping 10.0.0.5
PING 10.0.0.5 (10.0.0.5): 56 data bytes
36 bytes from 10.1.26.1: Time to live exceeded
Vr HL TOS  Len   ID Flg  off TTL Pro  cks      Src      Dst
```

```
 4  5  00 0054 e2db   0 0000  01  01 a8c6 10.1.26.2  10.0.0.5

36 bytes from 10.1.26.1: Time to live exceeded
Vr HL TOS  Len   ID Flg  off TTL Pro  cks      Src      Dst
 4  5  00 0054 e2de   0 0000  01  01 a8c3 10.1.26.2  10.0.0.5

36 bytes from 10.1.26.1: Time to live exceeded
Vr HL TOS  Len   ID Flg  off TTL Pro  cks      Src      Dst
 4  5  00 0054 e2e2   0 0000  01  01 a8bf 10.1.26.2  10.0.0.5

^C
--- 10.0.0.5 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss

user@R6> show route 10.0.0.5

inet.0: 20 destinations, 20 routes (20 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.5/32        *[IS-IS/165] 00:02:39, metric 10
                     > to 10.1.26.1  via so-0/0/2.0

user@R6> traceroute 10.0.0.5
traceroute to 10.0.0.5 (10.0.0.5), 30 hops max, 40 byte packets
 1  10.1.26.1 (10.1.26.1)  0.649 ms  0.521 ms  0.490 ms
 2  10.1.26.2 (10.1.26.2)  0.521 ms  0.537 ms  0.507 ms
 3  10.1.26.1 (10.1.26.1)  0.523 ms  0.536 ms  0.514 ms
 4  10.1.26.2 (10.1.26.2)  0.528 ms  0.551 ms  0.523 ms
 5  10.1.26.1 (10.1.26.1)  0.531 ms  0.550 ms  0.524 ms
```

Meaning

The sample output shows an unsuccessful `ping` command in which the packets are being rejected because the time to live is exceeded. The output for the `show route` command shows the interface (`10.1.26.1`) that you can examine further for possible problems. The `traceroute` command shows the loop between `10.1.26.1` (`R2`) and `10.1.26.2` (`R6`), as indicated by the continuous repetition of the two interface addresses.

Isolating the Causes of a Network Problem

IN THIS SECTION

Problem

Description

A particular symptom can be the result of one or more causes. Narrow down the focus of your search to find each individual cause of the unwanted behavior.

Solution

To isolate the cause of a particular problem, enter one or all of the following Junos OS CLI operational mode command:

```
             user@host> show < configuration | bgp | interfaces | isis | ospf | route
>
```

Your particular problem may require the use of more than just the commands listed above. See the appropriate command reference for a more exhaustive list of commonly used operational mode commands.

Sample Output

```
user@R6> show interfaces terse
Interface               Admin Link Proto Local                 Remote
so-0/0/0                up    up
so-0/0/0.0              up    up   inet  10.1.56.2/30
                                   iso
so-0/0/2                up    up
so-0/0/2.0              up    up   inet  10.1.26.2/30
                                   iso
so-0/0/3                up    up
```

```
so-0/0/3.0              up    up   inet  10.1.36.2/30
                                   iso
[...Output truncated...]
```

The following sample output is from R2:

```
user@R2> show route 10.0.0.5

inet.0: 22 destinations, 25 routes (22 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.5/32         *[Static/5] 00:16:21
                    > to 10.1.26.2 via so-0/0/2.0
                    [BGP/170] 3d 20:23:35, MED 5, localpref 100
                      AS path: 65001 I
                    > to 10.1.12.1 via so-0/0/0.0
```

Meaning

The sample output shows that all interfaces on R6 are up. The output from R2 shows that a static route [Static/5] configured on R2 points to R6 (10.1.26.2) and is the preferred route to R5 because of its low preference value. However, the route is looping from R2 to R6, as indicated by the missing reference to R5 (10.1.15.2).

Taking Appropriate Action for Resolving the Network Problem

IN THIS SECTION

Problem

Description

The appropriate action depends on the type of problem you have isolated. In this example, a static route configured on `R2` is deleted from the `[routing-options]` hierarchy level. Other appropriate actions might include the following:

Solution

- Check the local router's configuration and edit it if appropriate.
- Troubleshoot the intermediate router.
- Check the remote host configuration and edit it if appropriate.
- Troubleshoot routing protocols.
- Identify additional possible causes.

To resolve the problem in this example, enter the following Junos OS CLI commands:

```
[edit]
                          user@R2# delete routing-options static route destination-
prefix
user@R2# commit and-quit
user@R2# show route destination-prefix
```

Sample Output

```
[edit]
user@R2# delete routing-options static route 10.0.0.5/32

[edit]
user@R2# commit and-quit
commit complete
Exiting configuration mode

user@R2> show route 10.0.0.5

inet.0: 22 destinations, 24 routes (22 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
10.0.0.5/32        *[BGP/170] 3d 20:26:17, MED 5, localpref 100
                      AS path: 65001 I
                    > to 10.1.12.1 via so-0/0/0.0
```

Meaning

The sample output shows the static route deleted from the [routing-options] hierarchy and the new configuration committed. The output for the `show route` command now shows the BGP route as the preferred route, as indicated by the asterisk (*).

Evaluating the Solution to Check Whether the Network Problem Is Resolved

IN THIS SECTION

Problem

Description

If the problem is solved, you are finished. If the problem remains or a new problem is identified, start the process over again.

You can address possible causes in any order. In relation to the network in *Isolating a Broken Network Connection*, we chose to work from the local router toward the remote router, but you might start at a different point, particularly if you have reason to believe that the problem is related to a known issue, such as a recent change in configuration.

Solution

To evaluate the solution, enter the following Junos OS CLI commands:

```
                                       user@host> show route (ip-address   |host-name)
user@host> ping (ip-address  | host-name)
user@host> traceroute (ip-address  | host-name)
```

Sample Output

```
user@R6> show route 10.0.0.5

inet.0: 20 destinations, 20 routes (20 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.5/32        *[BGP/170]  00:01:35, MED 5, localpref 100, from 10.0.0.2
                      AS path: 65001 I
                    > to 10.1.26.1 via so-0/0/2.0

user@R6> ping 10.0.0.5
PING 10.0.0.5 (10.0.0.5): 56 data bytes
64 bytes from 10.0.0.5: icmp_seq=0 ttl=253 time=0.866 ms
64 bytes from 10.0.0.5: icmp_seq=1 ttl=253 time=0.837 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=253 time=0.796 ms
^C
--- 10.0.0.5 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.796/0.833/0.866/0.029 ms

user@R6> traceroute 10.0.0.5
traceroute to 10.0.0.5 (10.0.0.5), 30 hops max, 40 byte packets
 1  10.1.26.1 (10.1.26.1)  0.629 ms  0.538 ms  0.497 ms
 2  10.1.12.1 (10.1.12.1)  0.534 ms  0.538 ms  0.510 ms
 3  10.0.0.5 (10.0.0.5)  0.776 ms  0.705 ms  0.672 ms
```

Meaning

The sample output shows that there is now a connection between `R6` and `R5`. The `show route` command shows that the BGP route to `R5` is preferred, as indicated by the asterisk (*). The `ping` command is successful and the `traceroute` command shows that the path from `R6` to `R5` is through `R2` (`10.1.26.1`), and then through `R1` (`10.1.12.1`).

Checklist for Tracking Error Conditions

IN THIS SECTION

Problem

Description

Table 3 on page 168 provides links and commands for configuring routing protocol daemon tracing, Border Gateway Protocol (BGP), Intermediate System-to-Intermediate System (IS-IS) protocol, and Open Shortest Path First (OSPF) protocol tracing to diagnose error conditions.

Solution

Table 3: Checklist for Tracking Error Conditions

Tasks	Command or Action
Configure Routing Protocol Process Tracing	
1. *Configure Routing Protocol Process Tracing*	`[edit] edit routing-options traceoptions` `filename` `size` `size` files *number* show co log *filename*
1. *Configure Routing Protocol Tracing for a Specific Routing Protocol*	`[edit] edit protocol` `protocol-name` trace *filename* size *size* files *number* show com log *filename*
1. *Monitor Trace File Messages Written in Near-Real Time*	monitor start *filename*
1. *Stop Trace File Monitoring*	monitor stop *filename*

Table 3: Checklist for Tracking Error Conditions ***(Continued)***

Tasks	Command or Action
Configure BGP-Specific Options	
1. Display Detailed BGP Protocol Information	`[edit] edit protocol bgp traceoptions se` `detail show commit run show log filename`
1. Display Sent or Received BGP Packets	`[edit] edit protocol bgp traceoptions se` `(send | receive) show commit run show lo`
1. Diagnose BGP Session Establishment Problems	`[edit] edit protocol bgp set traceoption` `detail show commit run show log filename`
Configure IS-IS-Specific Options	
1. Displaying Detailed IS-IS Protocol Information	`[edit] edit protocol isis traceoptions s` `detail show commit run show log filename`
1. Displaying Sent or Received IS-IS Protocol Packets	`[edit] edit protocols isis traceoptions` `(send | receive) show commit run show lo`
1. Analyzing IS-IS Link-State PDUs in Detail	`[edit] edit protocols isis traceoptions` `detail show commit run show log filename`
Configure OSPF-Specific Options	
1. Diagnose OSPF Session Establishment Problems	`[edit] edit protocols ospf traceoptions` `detail show commit run show log filename`
1. Analyze OSPF Link-State Advertisement Packets in Detail	`[edit] edit protocols ospf traceoptions` `update detail show commit run show log`

Configure Routing Protocol Process Tracing

IN THIS SECTION

Action

To configure routing protocol process (rpd) tracing, follow these steps:

1. In configuration mode, go to the following hierarchy level:

```
[edit]
user@host# edit routing-options traceoptions
```

2. Configure the file, file size, number, and flags:

```
[edit routing-options traceoptions]
user@host#  set file filename size size file number
[edit routing-options traceoptions]
user@host#  set flag flag
```

For example:

```
[edit routing-options traceoptions]
user@host# set file daemonlog size 10240 files 10
[edit routing-options traceoptions]
user@host# set flag general
```

3. Verify the configuration:

```
user@host# show
```

For example:

```
[edit routing-options traceoptions]
user@host# show
file daemonlog size 10k files 10;
flag general;
```

4. Commit the configuration:

```
user@host# commit
```

NOTE: Some traceoptions flags generate an extensive amount of information. Tracing can also slow down the operation of routing protocols. Delete the traceoptions configuration if you no longer require it.

1. View the contents of the file containing the detailed messages:

```
user@host# run show log filename
```

For example:

```
[edit routing-options traceoptions]
user@pro4-a# run show log daemonlog
Sep 17 14:17:31 trace_on: Tracing to "/var/log/daemonlog" started
Sep 17 14:17:31 Tracing flags enabled: general
Sep 17 14:17:31 inet_routerid_notify: Router ID: 10.255.245.44
Sep 17 14:17:31 inet_routerid_notify: No Router ID assigned
Sep 17 14:17:31 Initializing LSI globals
Sep 17 14:17:31 LSI initialization complete
Sep 17 14:17:31 Initializing OSPF instances
Sep 17 14:17:31 Reinitializing OSPFv2 instance master
Sep 17 14:17:31 OSPFv2 instance master running
[...Output truncated...]
```

Meaning

Table 4 on page 172 lists tracing flags and example output for Junos-supported routing protocol daemon tracing.

Table 4: Routing Protocol Daemon Tracing Flags

Tracing Flag	Description	Example Output
all	All operations	Not available.
general	Normal operations and routing table change	Not available.
normal	Normal operations	Not available.
policy	Policy operations and actions	Nov 29 22:19:58 export: Dest 10.0.0.0 proto Static Nov 29 22:19:58 policy_match_qual_or: Qualifier proto Sense: 0 Nov 29 22:19:58 policy_match_qual_or: Qualifier proto Sense: 0 Nov 29 22:19:58 export: Dest 10.10.10.0 proto IS-IS
route	Routing table changes	Nov 29 22:23:59 Nov 29 22:23:59 rtlist_walker_job: rt_list walk for RIB inet.0 started with 42 entries Nov 29 22:23:59 rt_flash_update_callback: flash KRT (inet.0) start Nov 29 22:23:59 rt_flash_update_callback: flash KRT (inet.0) done Nov 29 22:23:59 rtlist_walker_job: rt_list walk for inet.0 ended with 42 entries Nov 29 22:23:59 Nov 29 22:23:59 KRT Request: send len 68 v14 seq 0 CHANGE route/user af 2 addr 172.16.0.0 nhop-type unicast nhop 10.10.10.33 Nov 29 22:23:59 KRT Request: send len 68 v14 seq 0 ADD route/user af 2 addr 172.17.0.0 nhop-type unicast nhop 10.10.10.33 Nov 29 22:23:59 KRT Request: send len 68 v14 seq 0 ADD route/user af 2 addr 10.149.3.0 nhop-type unicast nhop 10.10.10.33 Nov 29 22:24:19 trace_on: Tracing to "/var/log/rpdlog" started Nov 29 22:24:19 KRT Request: send len 68 v14 seq 0 DELETE route/user af 2 addr 10.10.218.0 nhop-type unicast nhop 10.10.10.29 Nov 29 22:24:19 RELEASE 10.10.218.0 255.255.255.0 gw 10.10.10.29,10.10.10.33 BGP pref 170/-101 metric so-1/1/0.0,so-1/1/1.0 <Release Delete Int Ext> as 65401 Nov 29 22:24:19 KRT Request: send len 68 v14 seq 0 DELETE route/user af 2 addr 172.18.0.0 nhop-type unicast nhop 10.10.10.33
state	State transitions	Not available.

Table 4: Routing Protocol Daemon Tracing Flags *(Continued)*

Tracing Flag	Description	Example Output
task	Interface transactions and processing	Nov 29 22:50:04 foreground dispatch running job task_collect for task Scheduler Nov 29 22:50:04 task_collect_job: freeing task MGMT_Listen (DELETED) Nov 29 22:50:04 foreground dispatch completed job task_collect for task Scheduler Nov 29 22:50:04 background dispatch running job rt_static_update for task RT Nov 29 22:50:04 task_job_delete: delete background job rt_static_update for task RT Nov 29 22:50:04 background dispatch completed job rt_static_update for task RT Nov 29 22:50:04 background dispatch running job Flash update for task RT Nov 29 22:50:04 background dispatch returned job Flash update for task RT Nov 29 22:50:04 background dispatch running job Flash update for task RT Nov 29 22:50:04 task_job_delete: delete background job Flash update for task RT Nov 29 22:50:04 background dispatch completed job Flash update for task RT Nov 29 22:50:04 background dispatch running job Flash update for task RT Nov 29 22:50:04 task_job_delete: delete background job Flash update for task RT
timer	Timer usage	Nov 29 22:52:07 task_timer_hiprio_dispatch: ran 1 timer Nov 29 22:52:07 main: running normal priority timer queue Nov 29 22:52:07 main: ran 1 timer Nov 29 22:52:07 task_timer_hiprio_dispatch: running high priority timer queue Nov 29 22:52:07 task_timer_hiprio_dispatch: ran 1 timer Nov 29 22:52:07 main: running normal priority timer queue Nov 29 22:52:07 main: ran 1 timer Nov 29 22:52:07 main: running normal priority timer queue Nov 29 22:52:07 main: ran 2 timers

Configure Routing Protocol Tracing for a Specific Routing Protocol

IN THIS SECTION

Action

To configure routing protocol tracing for a specific routing protocol, follow these steps:

1. In configuration mode, go to the following hierarchy level:

   ```
   [edit]
                    user@host# edit protocol protocol-name traceoptions
   ```

2. Configure the file, file size, number, and flags:

   ```
   [edit protocols protocol name traceoptions]
                    user@host# set file filename size size files
   number
   [edit protocols protocol name traceoptions]
   user@host# set flag flag
   ```

   For example:

   ```
   [edit protocols ospf traceoptions]
   user@host# set file ospflog size 10240 files 10
   [edit protocols ospf traceoptions]
   user@host# set flag general
   ```

3. Verify the configuration:

   ```
   user@host# show
   ```

   For example:

   ```
   [edit protocols ospf traceoptions]
   user@host# show
   file ospflog size 10k files 10;
   flag general;
   ```

4. Commit the configuration:

   ```
   user@host# commit
   ```

5. View the contents of the file containing the detailed messages:

```
user@host# run show log filename
```

For example:

```
[edit protocols ospf traceoptions]
user@pro4-a# run show log ospflog
Sep 17 14:23:10 trace_on: Tracing to "/var/log/ospflog" started
Sep 17 14:23:10 rt_flash_update_callback: flash OSPF (inet.0) start
Sep 17 14:23:10 OSPF:  multicast address 224.0.0.5/32, route ignored
Sep 17 14:23:10 rt_flash_update_callback: flash OSPF (inet.0) done
Sep 17 14:23:10 CHANGE   10.255.245.46/32   gw 10.10.208.67  OSPF     pref 10/0 metric 1/0
fe-0/0/0.0 <Delete Int>
Sep 17 14:23:10 CHANGE   10.255.245.46/32   gw 10.10.208.67  OSPF     pref 10/0 metric 1/0
fe-0/0/0.0 <Active Int>
Sep 17 14:23:10 ADD      10.255.245.46/32   gw 10.10.208.67  OSPF     pref 10/0 metric 1/0
fe-0/0/0.0 <Active Int>
Sep 17 14:23:10 CHANGE   10.255.245.48/32   gw 10.10.208.69  OSPF     pref 10/0 metric 1/0
fe-0/0/0.0 <Delete Int>
Sep 17 14:23:10 CHANGE   10.255.245.48/32   gw 10.10.208.69  OSPF     pref 10/0 metric 1/0
fe-0/0/0.0 <Active Int>
Sep 17 14:23:10 ADD      10.255.245.48/32   gw 10.10.208.69  OSPF     pref 10/0 metric 1/0
fe-0/0/0.0 <Active Int>
Sep 17 14:23:10 rt_close: 4/4 routes proto OSPF
[...Output truncated...]
```

Meaning

Table 5 on page 175 lists standard tracing options that are available globally or that can be applied to specific protocols. You can also configure tracing for a specific BGP peer or peer group. For more information, see the *Junos System Basics Configuration Guide*.

Table 5: Standard Trace Options for Routing Protocols

Tracing Flag	Description
all	All operations

Table 5: Standard Trace Options for Routing Protocols *(Continued)*

Tracing Flag	Description
general	Normal operations and routing table changes
normal	Normal operations
policy	Policy operations and actions
route	Routing table changes
state	State transitions
task	Interface transactions and processing
timer	Timer usage

Monitor Trace File Messages Written in Near-Real Time

IN THIS SECTION

Purpose

To monitor messages in near-real time as they are being written to a trace file.

Action

To monitor messages in near-real time as they are being written to a trace file, use the following Junos OS command-line interface (CLI) operational mode command:

```
user@host> monitor start  filename
```

Sample Output

command-name

```
user@host> monitor start isis
user@host>
*** isis ***
Sep 15 18:32:21 Updating LSP isis5.02-00 in database
Sep 15 18:32:21 Updating L2 LSP isis5.02-00 in TED
Sep 15 18:32:21 Adding a half link from isis5.02 to isis6.00
Sep 15 18:32:21 Adding a half link from isis5.02 to isis5.00
Sep 15 18:32:21 Adding a half link from isis5.02 to isis6.00
Sep 15 18:32:21 Adding a half link from isis5.02 to isis5.00
Sep 15 18:32:21 Scheduling L2 LSP isis5.02-00 sequence 0xd87 on interface fxp2.3
Sep 15 18:32:21 Updating LSP isis5.00-00 in database
Sep 15 18:32:21 Updating L1 LSP isis5.00-00 in TED
Sep 15 18:32:21 Sending L2 LSP isis5.02-00 on interface fxp2.3
Sep 15 18:32:21     sequence 0xd87, checksum 0xc1c8, lifetime 1200
```

Stop Trace File Monitoring

IN THIS SECTION

Action

To stop monitoring a trace file in near-real time, use the following Junos OS CLI operational mode command after you have started monitoring:

```
user@host                 monitor stop filename
```

Sample Output

```
user@host> monitor start isis
user@host>
*** isis ***
Sep 15 18:32:21 Updating LSP isis5.02-00 in database
Sep 15 18:32:21 Updating L2 LSP isis5.02-00 in TED
Sep 15 18:32:21 Adding a half link from isis5.02 to isis6.00
Sep 15 18:32:21 Adding a half link from isis5.02 to isis5.00
Sep 15 18:32:21 Adding a half link from isis5.02 to isis6.00
Sep 15 18:32:21 Adding a half link from isis5.02 to isis5.00
Sep 15 18:32:21 Scheduling L2 LSP isis5.02-00 sequence 0xd87 on interface fxp2.3
Sep 15 18:32:21 Updating LSP isis5.00-00 in database
Sep 15 18:32:21 Updating L1 LSP isis5.00-00 in TED
Sep 15 18:32:21 Sending L2 LSP isis5.02-00 on interface fxp2.3
Sep 15 18:32:21     sequence 0xd87, checksum 0xc1c8, lifetime 1200
monitor stop isis
user@host>
```

Tracing Global Routing Protocol Operations

IN THIS SECTION

Understanding Global Routing Protocol Tracing Operations

Global routing protocol tracing operations track all general routing operations and record them in a log file. To set protocol-specific tracing operations and to modify the global tracing operations for an individual protocol, configure tracing for that protocol.

Using the `traceoptions` statement, you can specify the following global routing protocol tracing flags:

- **all**—All tracing operations
- **condition-manager**—Condition manager events
- **config-internal**—Configuration internals
- **general**—All normal operations and routing table changes (a combination of the normal and route trace operations)
- **graceful-restart**—Graceful restart operations
- **normal**—All normal operations
- **nsr-synchronization**—Nonstop routing synchronization events
- **parse**—Configuration parsing
- **policy**—Policy operations and actions
- **regex-parse**—Regular expression parsing
- **route**—Routing table changes
- **state**—State transitions
- **task**—Interface transactions and processing
- **timer**—Timer usage

NOTE: Use the **all** flag with caution. This flag might cause the CPU to become very busy.

SEE ALSO

Junos OS Administration Library for Routing Devices

Example: Tracing Global Routing Protocol Operations

IN THIS SECTION

This example shows how to list and view files that are created when you enable global routing trace operations.

Requirements

You must have the **view** privilege.

Overview

To configure global routing protocol tracing, include the `traceoptions` statement at the `[edit routing-options]` hierarchy level:

```
traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <disable>;
}
```

The flags in a `traceoptions flag` statement are identifiers. When you use the `set` command to configure a flag, any flags that might already be set are not modified. In the following example, setting the **timer** tracing flag has no effect on the already configured **task** flag. Use the `delete` command to delete a particular flag.

```
[edit routing-options traceoptions]
user@host# show
flag task;
user@host# set traceoptions flag timer
user@host# show
flag task;
flag timer;
user@host# delete traceoptions flag task
user@host# show
flag timer;
```

This example shows how to configure and view a trace file that tracks changes in the routing table. The steps can be adapted to apply to trace operations for any Junos OS hierarchy level that supports trace operations.

TIP: To view a list of hierarchy levels that support tracing operations, enter the `help apropos traceoptions` command in configuration mode.

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

```
set routing-options traceoptions file routing-table-changes
set routing-options traceoptions file size 10m
set routing-options traceoptions file files 10
set routing-options traceoptions flag route
set routing-options static route 1.1.1.2/32 next-hop 10.0.45.6
```

Configuring Trace Operations

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure the trace operations:

1. Configure trace operations.

```
[edit routing-options traceoptions]
user@host# set file routing-table-changes
user@host# set file size 10m
user@host# set file files 10
user@host# set flag route
```

2. Configure a static route to cause a change in the routing table.

```
[edit routing-options static]
user@host# set route 1.1.1.2/32 next-hop 10.0.45.6
```

3. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Viewing the Trace File

Step-by-Step Procedure

To view the trace file:

1. In operational mode, list the log files on the system.

```
user@host> file list /var/log
/var/log:
...
```

```
routing-table-changes
...
```

2. View the contents of the **routing-table-changes** file.

```
user@host> file show /var/log/routing-table-changes
Dec 15 11:09:29 trace_on: Tracing to "/var/log/routing-table-changes" started
Dec 15 11:09:29.496507
Dec 15 11:09:29.496507 Tracing flags enabled: route
Dec 15 11:09:29.496507
Dec 15 11:09:29.533203 inet_routerid_notify: Router ID: 192.168.4.1
Dec 15 11:09:29.533334 inet_routerid_notify: No Router ID assigned
Dec 15 11:09:29.533381 inet_routerid_notify: No Router ID assigned
Dec 15 11:09:29.533420 inet_routerid_notify: No Router ID assigned
Dec 15 11:09:29.534915 inet_routerid_notify: Router ID: 192.168.4.1
Dec 15 11:09:29.542934 inet_routerid_notify: No Router ID assigned
Dec 15 11:09:29.549253 inet_routerid_notify: No Router ID assigned
Dec 15 11:09:29.556878 inet_routerid_notify: No Router ID assigned
Dec 15 11:09:29.582990 rt_static_reinit: examined 3 static nexthops, 0 unreferenced
Dec 15 11:09:29.589920
Dec 15 11:09:29.589920 task_reconfigure reinitializing done
...
```

3. Filter the output of the log file.

```
user@host> file show /var/log/routing-table-changes | match 1.1.1.2
Dec 15 11:15:30.780314 ADD      1.1.1.2/32          nhid 0 gw 10.0.45.6       Static   pref
5/0 metric  at-0/2/0.0 <ctive Int Ext>
Dec 15 11:15:30.782276 KRT Request: send len 216 v104 seq 0 ADD route/user af 2 table 0 infot
0 addr 1.1.1.2 nhop-type unicast nhindex 663
```

4. View the tracing operations in real time by running the `monitor start` command with an optional **match** condition.

```
user@host> monitor start routing-table-changes | match 1.1.1.2
Aug 10 19:21:40.773467 BGP RECV         0.0.0.0/0
Aug 10 19:21:40.773685 bgp_rcv_nlri: 0.0.0.0/0
Aug 10 19:21:40.773778 bgp_rcv_nlri: 0.0.0.0/0 belongs to meshgroup
Aug 10 19:21:40.773832 bgp_rcv_nlri: 0.0.0.0/0 qualified bnp->ribact 0x0 l2afcb 0x0
```

5. Deactivate the static route.

```
user@host# deactivate routing-options static route 1.1.1.2/32
user@host# commit
```

```
*** routing-table-changes ***
Dec 15 11:42:59.355557 CHANGE   1.1.1.2/32          nhid 663 gw 10.0.45.6       Static   pref
5/0 metric  at-0/2/0.0 <Delete Int Ext>
Dec 15 11:42:59.426887 KRT Request: send len 216 v104 seq 0 DELETE route/user af 2 table 0
infot 0 addr 1.1.1.2 nhop-type discard filtidx 0
Dec 15 11:42:59.427366 RELEASE  1.1.1.2/32          nhid 663 gw 10.0.45.6       Static   pref
5/0 metric  at-0/2/0.0 <Release Delete Int Ext>
```

6. Halt the `monitor` command by pressing Enter and typing **monitor stop**.

```
[Enter]
user@host> monitor stop
```

7. When you are finished troubleshooting, consider deactivating trace logging to avoid any unnecessary impact to system resources.

   When configuration is deactivated, it appears in the configuration with the **inactive** tag.

```
[edit routing-options]
user@host# deactivate traceoptions
user@host# commit
```

```
[edit routing-options]
user@host# show

inactive: traceoptions {
    file routing-table-changes size 10m files 10;
    flag route;
}
static {
    inactive: route 1.1.1.2/32 next-hop 10.0.45.6;
}
```

8. To reactivate trace operations, use the **activate** configuration-mode statement.

```
[edit routing-options]
user@host# activate traceoptions
user@host# commit
```

Results

From configuration mode, confirm your configuration by entering the `show routing-options` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show routing-options
traceoptions {
    file routing-table-changes size 10m files 10;
    flag route;
}
static {
    route 1.1.1.2/32 next-hop 10.0.45.6;
}
```

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying That the Trace Log File Is Operating

Purpose

Make sure that events are being written to the log file.

Action

```
user@host> **show log routing-table-changes**
Dec 15 11:09:29 trace_on: Tracing to "/var/log/routing-table-changes" started
```

9

CHAPTER

Configuration Statements and Operational Commands

IN THIS CHAPTER

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Read this guide to learn about the syntax and options that make up the statements and commands. Also understand the contexts in which you'll use these CLI elements in your network configurations and operations.

- Junos CLI Reference

Click the links to access Junos OS and Junos OS Evolved configuration statement and command summary topics.

- Configuration Statements
- Operational Commands