

**Junos® OS**

---

# IPv6 Neighbor Discovery User Guide

Published  
2025-06-24

Juniper Networks, Inc.  
1133 Innovation Way  
Sunnyvale, California 94089  
USA  
408-745-2000  
[www.juniper.net](http://www.juniper.net)

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

*Junos® OS IPv6 Neighbor Discovery User Guide*  
Copyright © 2025 Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

## YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

## END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at <https://support.juniper.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

# Table of Contents

[About This Guide | v](#)

1

## [Configuring IPv6 Neighbor Discovery](#)

### [IPv6 Neighbor Discovery | 2](#)

[IPv6 Neighbor Discovery Overview | 2](#)

[Supported ICMP Router Discovery and IPv6 Neighbor Discovery Standards | 6](#)

[Example: Configuring IPv6 Interfaces and Enabling Neighbor Discovery | 6](#)

[Requirements | 6](#)

[Overview | 7](#)

[Configuration | 9](#)

[Verification | 13](#)

### [Secure IPv6 Neighbor Discovery | 19](#)

[Understanding Secure IPv6 Neighbor Discovery | 19](#)

[Example: Configuring Secure IPv6 Neighbor Discovery | 20](#)

[Requirements | 20](#)

[Overview | 20](#)

[Configuration | 22](#)

[Verification | 24](#)

### [NDP Proxy and DAD Proxy | 26](#)

[Overview | 26](#)

[Configuring NDP Proxy | 28](#)

[Configuring DAD Proxy | 29](#)

[NDP Proxy Support For User Routes | 30](#)

### [Neighbor Discovery Cache Protection | 31](#)

[Neighbor Discovery Cache Protection Overview | 32](#)

[Configuring Neighbor Discovery Cache Protection | 33](#)

Example: Configuring Neighbor Discovery Cache Protection to Prevent Denial-of-Service Attacks | 34

Requirements | 35

Overview | 35

Configuration | 36

Verification | 37

## Router Advertisement Proxy | 40

Overview | 41

Configure RA Proxy | 42

## 2

## Troubleshooting

### Troubleshooting Network Issues | 45

Working with Problems on Your Network | 45

Isolating a Broken Network Connection | 46

Identifying the Symptoms of a Broken Network Connection | 48

Isolating the Causes of a Network Problem | 50

Taking Appropriate Action for Resolving the Network Problem | 51

Evaluating the Solution to Check Whether the Network Problem Is Resolved | 53

Checklist for Tracking Error Conditions | 55

Configure Routing Protocol Process Tracing | 57

Configure Routing Protocol Tracing for a Specific Routing Protocol | 60

Monitor Trace File Messages Written in Near-Real Time | 63

Stop Trace File Monitoring | 64

## 3

## Configuration Statements and Operational Commands

### Junos CLI Reference Overview | 67

# About This Guide

Use this guide to configure, monitor, and troubleshoot the IPv6 neighbor discovery on your Juniper Network devices.

## RELATED DOCUMENTATION

| [Day One: Exploring IPv6](#)

# 1

CHAPTER

## Configuring IPv6 Neighbor Discovery

---

### IN THIS CHAPTER

- IPv6 Neighbor Discovery | 2
  - Secure IPv6 Neighbor Discovery | 19
  - NDP Proxy and DAD Proxy | 26
  - Neighbor Discovery Cache Protection | 31
  - Router Advertisement Proxy | 40
-

# IPv6 Neighbor Discovery

## SUMMARY

Neighbor discovery is a protocol used for IPv6 traffic that allows different nodes on the same link to advertise their existence to their neighbors, and to learn about the existence of their neighbors.

## IN THIS SECTION

- [IPv6 Neighbor Discovery Overview | 2](#)
- [Supported ICMP Router Discovery and IPv6 Neighbor Discovery Standards | 6](#)
- [Example: Configuring IPv6 Interfaces and Enabling Neighbor Discovery | 6](#)

## IPv6 Neighbor Discovery Overview

### IN THIS SECTION

- [Improvements Over IPv4 Protocols | 3](#)
- [Router Discovery | 4](#)
- [Address Resolution | 4](#)
- [Redirect | 4](#)
- [SLAAC | 5](#)

Neighbor discovery is a protocol that allows different nodes on the same link to advertise their existence to their neighbors, and to learn about the existence of their neighbors.

Routers and hosts (nodes) use Neighbor Discovery (ND) messages to determine the link-layer addresses of neighbors that reside on attached links and to overwrite invalid cache entries. Hosts also use ND to find neighboring routers that can forward packets on their behalf.

In addition, nodes use ND to actively track the ability to reach neighbors. When a router (or the path to a router) fails, nodes actively search for alternatives to reach the destination.

This section discusses the following topics:

## Improvements Over IPv4 Protocols

IPv6 Neighbor Discovery corresponds to a number of the IPv4 protocols — ARP, ICMP Router Discovery, and ICMP Redirect. However, Neighbor Discovery provides many improvements over the IPv4 set of protocols. These improvements address the following:

- Router discovery—How a host locates routers residing on an attached link.
- Prefix discovery—How a host discovers address prefixes for destinations residing on an attached link. Nodes use prefixes to distinguish between destinations that reside on an attached link and those destinations that it can reach only through a router.
- Parameter discovery—How a node learns various parameters (link parameters or Internet parameters) that it places in outgoing packets.
- Address resolution—How a node uses only a destination IPv6 address to determine a link-layer address for destinations on an attached link.
- Next-hop determination—The algorithm that a node uses for mapping an IPv6 destination address into a neighbor IPv6 address (either the next router hop or the destination itself) to which it plans to send traffic for the destination.
- Neighbor unreachability detection—How a node determines that it can no longer reach a neighbor.
- Duplicate address detection—How a node determines whether an address is already in use by another node.

A router periodically multicasts a router advertisement from each of its multicast interfaces, announcing its availability. Hosts listen for these advertisements for address autoconfiguration and discovery of link-local addresses of the neighboring routers. When a host starts, it multicasts a router solicitation to ask for immediate advertisements.

The router discovery messages do not constitute a routing protocol. They enable hosts to discover the existence of neighboring routers, but are not used to determine which router is best to reach a particular destination.

Neighbor discovery uses the following Internet Control Message Protocol version 6 (ICMPv6) messages: router solicitation, router advertisement, neighbor solicitation, neighbor advertisement, and redirect.

Neighbor discovery for IPv6 replaces the following IPv4 protocols: router discovery (RDISC), Address Resolution Protocol (ARP), and ICMPv4 redirect.

Junos OS Release 9.3 and later supports Secure Neighbor Discovery (SEND). SEND enables you to secure Neighbor Discovery protocol (NDP) messages. It is applicable in environments where physical security on a link is not assured and attacks on NDP messages are a concern. The Junos OS secures NDP messages through cryptographically generated addresses (CGAs).



## Router Discovery

Router advertisements can contain a list of prefixes. These prefixes are used for address autoconfiguration, to maintain a database of onlink (on the same data link) prefixes, and for duplication address detection. If a node is onlink, the router forwards packets to that node. If the node is not onlink, the packets are sent to the next router for consideration. For IPv6, each prefix in the prefix list can contain a prefix length, a valid lifetime for the prefix, a preferred lifetime for the prefix, an onlink flag, and an autoconfiguration flag. This information enables address autoconfiguration and the setting of link parameters such as maximum transmission unit (MTU) size and hop limit.

Junos OS Release 22.4R1 and later supports NAT64 IPv6 address prefix router advertisement. The router advertises the configured NAT64 IPv6 address prefix in the router advertisement packets. You can configure up to 3 NAT64 IPv6 address prefix per interface.

You can configure the NAT64 IPv6 address prefix using the command `set protocols router-advertisement interface <interface-name> nat-prefix <prefix>`.

You can configure the router advertisement time using the command `set protocols router-advertisement interface <interface-name> nat-prefix <prefix> lifetime <lifetime>`.

## Address Resolution

For IPv6, ICMPv6 neighbor discovery replaces Address Resolution Protocol (ARP) for resolving network addresses to link-level addresses. Neighbor discovery also handles changes in link-layer addresses, inbound load balancing, anycast addresses, and proxy advertisements.

Nodes requesting the link-layer address of a target node multicast a neighbor solicitation message with the target address. The target sends back a neighbor advertisement message containing its link-layer address.

Neighbor solicitation and advertisement messages are used for detecting duplicate unicast addresses on the same link. Autoconfiguration of an IP address depends on whether there is a duplicate address on that link. Duplicate address detection is a requirement for autoconfiguration.

Neighbor solicitation and advertisement messages are also used for neighbor unreachability detection. Neighbor unreachability detection involves detecting the presence of a target node on a given link.

## Redirect

Redirect messages are sent to inform a host of a better next-hop router to a particular destination or an onlink neighbor. This is similar to ICMPv4 redirect. Very similar to the ICMPv4 Redirect feature, the ICMPv6 redirect message is used by routers to inform on-link hosts of a better next-hop for a given destination. The intent is to allow the routers to help hosts make the most efficient local routing decisions possible.

## SLAAC

In addition to all the other improvements it brings to the networking world, Neighbor Discovery also enables address autoconfiguration, namely Stateless Address Autoconfiguration (SLAAC). IPv6 maintains the capability for stateful address assignment through DHCPv6 (and static assignment), but SLAAC provides a lightweight address configuration method that might be desirable in many circumstances.

SLAAC provides plug-and-play IP connectivity in two phases: Phase 1: Link-local address assignment; and then, in Phase 2: Global address assignment.

- Phase 1—Steps for local connectivity:
  1. **Link-Local Address Generation:** Any time that a multicast-capable IPv6-enabled interface is turned up, the node generates a link-local address for that interface. This is done by appending an interface identifier to the link-local prefix (FE80::/10). The auto generated link-local address cannot be deleted. However, a new link-local address can also be manually entered, which overwrites the auto generated link-local address.
  2. **Duplicate Detection:** Before assigning the new link-local address to its interface, the node verifies that the address is unique. This is accomplished by sending a Neighbor Solicitation message destined to the new address. If there is a reply, then the address is a duplicate and the process stops, requiring operator intervention.
  3. **Link-Local Address Assignment:** If the address is unique, the node assigns it to the interface for which it was generated.

At this point, the node has IPv6 connectivity to all other nodes on the same link. Phase 2 can only be completed by hosts. The router's interface addresses must be configured by other means.

- Phase 2—Steps for global connectivity:
  1. **Router Advertisement:** The node sends a Router Solicitation to prompt all on-link routers to send it router advertisements. When the router is enabled to provide stateless autoconfiguration support, the router advertisement contains a subnet prefix for use by neighboring hosts.
  2. **Global Address Generation:** Once it receives a subnet prefix from a router, the host generates a global address by appending the interface id to the supplied prefix.
  3. **Duplicate Address Detection:** The host again performs Duplicate Address Detection (DAD), this time for the new global address.
  4. **Global Address Assignment:** Assuming that the address is not a duplicate, the host assigns it to the interface.

This process ensures full IPv6 global connectivity with no manual host configuration and very little router configuration.

## Supported ICMP Router Discovery and IPv6 Neighbor Discovery Standards

Junos OS substantially supports the following RFCs, which define standards for the Internet Control Message Protocol (ICMP for IP version 4 [IPv4]) and neighbor discovery (for IP version 6 [IPv6]).

- RFC 1256, *ICMP Router Discovery Messages*
- RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*
- RFC 2463, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*
- RFC 4443, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*
- RFC 4862, *IPv6 Stateless Address Autoconfiguration*
- RFC 8335, *PROBE: A Utility for Probing Interfaces*

## Example: Configuring IPv6 Interfaces and Enabling Neighbor Discovery

### IN THIS SECTION

- [Requirements | 6](#)
- [Overview | 7](#)
- [Configuration | 9](#)
- [Verification | 13](#)

This example shows how to configure the router or switch to send IPv6 neighbor discovery messages.

### Requirements

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

## Overview

### IN THIS SECTION

- Topology | 8

In this example, all of the interfaces in the sample topology are configured with IPv6 addresses. If you plan to extend IPv6 functionality into your LAN, datacenter, or customer networks, you might want to use Stateless Address Auto-Configuration (SLAAC) and that means configuring router advertisements. SLAAC is an IPv6 protocol that provides some similar functionality to DHCP in IPv4. Using SLAAC, network hosts can autoconfigure a globally unique IPv6 address based on the prefix provided by a nearby router in a router advertisement. This removes the need to explicitly configure every interface in a given section of the network. Router advertisement messages are disabled by default, and you must enable them to take advantage of SLAAC.

To configure the router to send router advertisement messages, you must include at least the following statements in the configuration. All other router advertisement configuration statements are optional.

```
protocols {
  router-advertisement {
    interface interface-name {
      prefix prefix;
    }
  }
}
```

To configure neighbor discovery, include the following statements. You configure router advertisement on a per-interface basis.

```
protocols {
  router-advertisement {
    interface interface-name {
      current-hop-limit number;
      default-lifetime seconds;
      (link-mtu | no-link-mtu);
      (managed-configuration | no-managed-configuration);
    }
  }
}
```

```

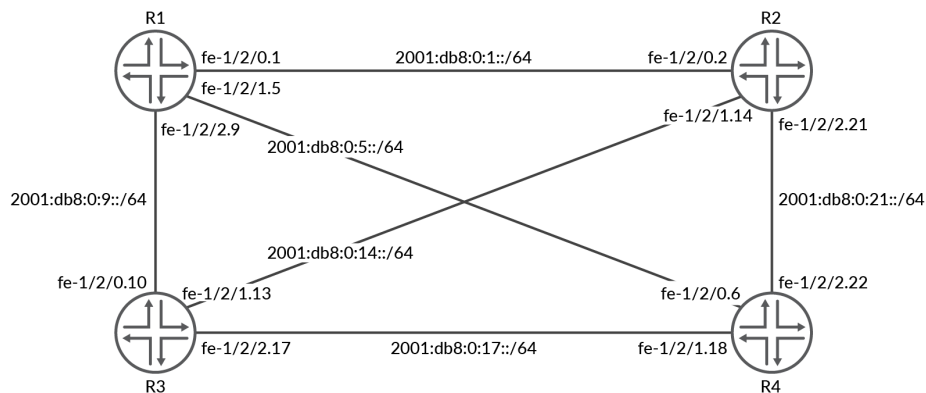
max-advertisement-interval seconds;
min-advertisement-interval seconds;
(other-stateful-configuration | no-other-stateful-configuration);
prefix prefix {
    (autonomous | no-autonomous);
    (on-link | no-on-link);
    preferred-lifetime seconds;
    valid-lifetime seconds;
}
reachable-time milliseconds;
retransmit-timer milliseconds;
solicit-router-advertisement-unicast;
virtual-router-only;
}
traceoptions {
    file filename <files number> <size maximum-file-size> <world-readable | no-world-
readable>;
    flag flag;
}
}
}

```

## Topology

Figure 1 on page 8 shows a simplified sample topology.

Figure 1: ICMP Router Discover Topology



This example shows how to make sure that all of the IPv6 hosts attached to the subnets in the sample topology can auto-configure a local EUI-64 address.

"CLI Quick Configuration" on page 9 shows the configuration for all of the devices in [Figure 1 on page 8](#). "No Link Title" on page 10 describes the steps on Device R1.

## Configuration

### IN THIS SECTION

- [Procedure | 9](#)

## 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 R1

```
set interfaces fe-1/2/0 unit 1 description to-P2
set interfaces fe-1/2/0 unit 1 family inet6 address 2001:db8:0:1::/64 eui-64
set interfaces fe-1/2/1 unit 5 description to-P4
set interfaces fe-1/2/1 unit 5 family inet6 address 2001:db8:0:5::/64 eui-64
set interfaces fe-1/2/2 unit 9 description to-P3
set interfaces fe-1/2/2 unit 9 family inet6 address 2001:db8:0:9::/64 eui-64
set interfaces lo0 unit 1 family inet6 address 2001:db8::1/128
set protocols router-advertisement interface fe-1/2/0.1 prefix 2001:db8:0:1::/64
set protocols router-advertisement interface fe-1/2/1.5 prefix 2001:db8:0:5::/64
set protocols router-advertisement interface fe-1/2/2.9 prefix 2001:db8:0:9::/64
```

#### Device R2

```
set interfaces fe-1/2/0 unit 2 description to-P1
set interfaces fe-1/2/0 unit 2 family inet6 address 2001:db8:0:1::/64 eui-64
set interfaces fe-1/2/1 unit 14 description to-P3
set interfaces fe-1/2/1 unit 14 family inet6 address 2001:db8:0:14::/64 eui-64
```

```

set interfaces fe-1/2/2 unit 21 description to-P4
set interfaces fe-1/2/2 unit 21 family inet6 address 2001:db8:0:21::/64 eui-64
set interfaces lo0 unit 2 family inet6 address 2001:db8::2/128
set protocols router-advertisement interface fe-1/2/0.2 prefix 2001:db8:0:1::/64
set protocols router-advertisement interface fe-1/2/1.14 prefix 2001:db8:0:14::/64
set protocols router-advertisement interface fe-1/2/2.21 prefix 2001:db8:0:21::/64

```

### Device R3

```

set interfaces fe-1/2/0 unit 10 description to-P1
set interfaces fe-1/2/0 unit 10 family inet6 address 2001:db8:0:9::/64 eui-64
set interfaces fe-1/2/1 unit 13 description to-P2
set interfaces fe-1/2/1 unit 13 family inet6 address 2001:db8:0:14::/64 eui-64
set interfaces fe-1/2/2 unit 17 description to-P4
set interfaces fe-1/2/2 unit 17 family inet6 address 2001:db8:0:17::/64 eui-64
set interfaces lo0 unit 3 family inet6 address 2001:db8::3/128
set protocols router-advertisement interface fe-1/2/0.10 prefix 2001:db8:0:9::/64
set protocols router-advertisement interface fe-1/2/1.13 prefix 2001:db8:0:14::/64
set protocols router-advertisement interface fe-1/2/2.17 prefix 2001:db8:0:17::/64

```

### Device R4

```

set interfaces fe-1/2/0 unit 6 description to-P1
set interfaces fe-1/2/0 unit 6 family inet6 address 2001:db8:0:5::/64 eui-64
set interfaces fe-1/2/1 unit 18 description to-P3
set interfaces fe-1/2/1 unit 18 family inet6 address 2001:db8:0:17::/64 eui-64
set interfaces fe-1/2/2 unit 22 description to-P2
set interfaces fe-1/2/2 unit 22 family inet6 address 2001:db8:0:21::/64 eui-64
set interfaces lo0 unit 4 family inet6 address 2001:db8::4/128
set protocols router-advertisement interface fe-1/2/0.6 prefix 2001:db8:0:5::/64
set protocols router-advertisement interface fe-1/2/1.18 prefix 2001:db8:0:17::/64
set protocols router-advertisement interface fe-1/2/2.22 prefix 2001:db8:0:21::/64

```

## 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 [CLI User Guide](#).

To configure a IPv6 neighbor discovery:

1. Configure the network interfaces.

This example shows multiple loopback interface addresses to simulate attached networks.

```
[edit interfaces]
user@R1# set fe-1/2/0 unit 1 description to-P2
user@R1# set fe-1/2/0 unit 1 family inet6 address 2001:db8:0:1::/64 eui-64
user@R1# set fe-1/2/1 unit 5 description to-P4
user@R1# set fe-1/2/1 unit 5 family inet6 address 2001:db8:0:5::/64 eui-64
user@R1# set fe-1/2/2 unit 9 description to-P3
user@R1# set fe-1/2/2 unit 9 family inet6 address 2001:db8:0:9::/64 eui-64
user@R1# set lo0 unit 1 family inet6 address 2001:db8::1/128
```

## 2. Enable neighbor discovery.

```
[edit protocols router-advertisement]
user@R1# set interface fe-1/2/0.1 prefix 2001:db8:0:1::/64
user@R1# set interface fe-1/2/1.5 prefix 2001:db8:0:5::/64
user@R1# set interface fe-1/2/2.9 prefix 2001:db8:0:9::/64
```

## Results

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

```
user@R1# show interfaces
fe-1/2/0 {
  unit 1 {
    description to-P2;
    family inet6 {
      address 2001:db8:0:1::/64 {
        eui-64;
      }
    }
  }
}
fe-1/2/1 {
  unit 5 {
    description to-P4;
    family inet6 {
      address 2001:db8:0:5::/64 {
```



```

        eui-64;
    }
}
}
fe-1/2/2 {
    unit 9 {
        description to-P3;
        family inet6 {
            address 2001:db8:0:9::/64 {
                eui-64;
            }
        }
    }
}
lo0 {
    unit 1 {
        family inet6 {
            address 2001:db8::1/128;
        }
    }
}
}

```

```

user@R1# show protocols
router-advertisement {
    interface fe-1/2/0.1 {
        prefix 2001:db8:0:1::/64;
    }
    interface fe-1/2/1.5 {
        prefix 2001:db8:0:5::/64;
    }
    interface fe-1/2/2.9 {
        prefix 2001:db8:0:9::/64;
    }
}

```

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

## Verification

### IN THIS SECTION

- [Checking the Interfaces | 13](#)
- [Pinging the Interfaces | 14](#)
- [Checking the IPv6 Neighbor Cache | 15](#)
- [Verifying IPv6 Router Advertisements | 15](#)
- [Tracing Neighbor Discovery Events | 17](#)

To confirm that the configuration is working properly, perform this task:

### Checking the Interfaces

#### Purpose

Verify that the interfaces are up, and view the assigned EUI-64 addresses.

#### Action

From operational mode, enter the `show interfaces terse` command.

```
user@R1> show interfaces terse
```

Interface	Admin	Link	Proto	Local	Remote
fe-1/2/0					
fe-1/2/0.1	up	up	inet6	2001:db8:0:1:2a0:a514:0:14c/64 fe80::2a0:a514:0:14c/64	
fe-1/2/1.5	up	up	inet6	2001:db8:0:5:2a0:a514:0:54c/64 fe80::2a0:a514:0:54c/64	
fe-1/2/2.9	up	up	inet6	2001:db8:0:9:2a0:a514:0:94c/64 fe80::2a0:a514:0:94c/64	
lo0					
lo0.1	up	up	inet6	2001:db8::1 fe80::2a0:a50f:fc56:14c	

## Meaning

The output shows that all interfaces are configured with the IPv6 (inet6) address family. Each IPv6-enabled interface has two IPv6 addresses; one link-local address, and one global address. The global addresses match those shown in [Figure 1 on page 8](#). Junos OS automatically creates a link-local address for any interface that is enabled for IPv6 operation. All link-local addresses begin with the fe80::/64 prefix. The host portion of the address is a full 64 bits long and matches the link-local interface identifier. When an interface address is configured using the eui-64 statement, its interface identifier matches the interface identifier of the link-local address. This is because link-local addresses are coded according to the EUI-64 specification.

## Pinging the Interfaces

### Purpose

Verify connectivity between the directly connected interfaces.

### Action

1. Determine the remote router's IPv6 interface address.

On Device R2, run the `show interfaces terse` command for the interface that is directly connected to Device R1, and copy the global address into the capture buffer of your terminal emulator.

```
user@R2> show interfaces fe-1/2/0.2 terse
```

Interface	Admin	Link	Proto	Local	Remote
fe-1/2/0.2	up	up	inet6	2001:db8:0:1:2a0:a514:0:24c/64	
				fe80::2a0:a514:0:24c/64	

2. On Device R1, run the `ping` command, using the global address that you copied.

```
user@R1> ping 2001:db8:0:1:2a0:a514:0:24c
PING6(56=40+8+8 bytes) 2001:db8:0:1:2a0:a514:0:14c --> 2001:db8:0:1:2a0:a514:0:24c
16 bytes from 2001:db8:0:1:2a0:a514:0:24c, icmp_seq=0 hlim=64 time=20.412 ms
16 bytes from 2001:db8:0:1:2a0:a514:0:24c, icmp_seq=1 hlim=64 time=18.897 ms
16 bytes from 2001:db8:0:1:2a0:a514:0:24c, icmp_seq=2 hlim=64 time=1.389 ms
```

## Meaning

Junos OS uses the same ping command for both IPv4 and IPv6 testing. The lack of any interior gateway protocol (IGP) in the network limits the ping testing to directly-connected neighbors. Repeat the ping test for other directly connected neighbors.

## Checking the IPv6 Neighbor Cache

### Purpose

Display information about the IPv6 neighbors.

After conducting ping testing, you can find an entries for interface addresses in the IPv6 neighbor cache.

### Action

From operational mode, enter the `show ipv6 neighbors` command.

```
user@R1> show ipv6 neighbors
```

IPv6 Address	Linklayer Address	State	Exp	Rtr	Secure	Interface
2001:db8:0:1:2a0:a514:0:24c	00:05:85:8f:c8:bd	stale	546	yes	no	fe-1/2/0.1
fe80::2a0:a514:0:24c	00:05:85:8f:c8:bd	stale	258	yes	no	fe-1/2/0.1
fe80::2a0:a514:0:64c	00:05:85:8f:c8:bd	stale	111	yes	no	fe-1/2/1.5
fe80::2a0:a514:0:a4c	00:05:85:8f:c8:bd	stale	327	yes	no	fe-1/2/2.9

## Meaning

In IPv6, the Address Resolution Protocol (ARP) has been replaced by the Neighbor Discovery Protocol (NDP). The IPv4 command `show arp` is replaced by the IPv6 command `show ipv6 neighbors`. The key pieces of information displayed by this command are the IP address, the MAC (Link Layer) address, and the interface.

## Verifying IPv6 Router Advertisements

### Purpose

Confirm that devices can be added to the network using SLAAC by ensuring that router advertisements are working properly.

## Action

From operational mode, enter the `show ipv6 router-advertisement` command.

```
user@R1> show ipv6 router-advertisement
Interface: fe-1/2/0.1
  Advertisements sent: 37, last sent 00:01:41 ago
  Solicits received: 0
  Advertisements received: 38
  Advertisement from fe80::2a0:a514:0:24c, heard 00:05:46 ago
    Managed: 0
    Other configuration: 0
    Reachable time: 0 ms
    Default lifetime: 1800 sec
    Retransmit timer: 0 ms
    Current hop limit: 64
    Prefix: 2001:db8:0:1::/64
      Valid lifetime: 2592000 sec
      Preferred lifetime: 604800 sec
    On link: 1
    Autonomous: 1
Interface: fe-1/2/1.5
  Advertisements sent: 36, last sent 00:05:49 ago
  Solicits received: 0
  Advertisements received: 37
  Advertisement from fe80::2a0:a514:0:64c, heard 00:00:54 ago
    Managed: 0
    Other configuration: 0
    Reachable time: 0 ms
    Default lifetime: 1800 sec
    Retransmit timer: 0 ms
    Current hop limit: 64
    Prefix: 2001:db8:0:5::/64
      Valid lifetime: 2592000 sec
      Preferred lifetime: 604800 sec
    On link: 1
    Autonomous: 1
Interface: fe-1/2/2.9
  Advertisements sent: 36, last sent 00:01:37 ago
  Solicits received: 0
  Advertisements received: 38
  Advertisement from fe80::2a0:a514:0:a4c, heard 00:01:00 ago
```

```

Managed: 0
Other configuration: 0
Reachable time: 0 ms
Default lifetime: 1800 sec
Retransmit timer: 0 ms
Current hop limit: 64
Prefix: 2001:db8:0:9::/64
  Valid lifetime: 2592000 sec
  Preferred lifetime: 604800 sec
On link: 1
Autonomous: 1

```

## Meaning

The output shows that router advertisements are being sent and received on Device R1's interfaces, indicating that both Device R1 and its directly connected neighbors are configured to generate router-advertisements.

## Tracing Neighbor Discovery Events

### Purpose

Perform additional validation by tracing router advertisements.

### Action

1. Configure trace operations.

```

[edit protocols router-advertisement traceoptions]
user@R1# set file ipv6-nd-trace
user@R1# set traceoptions flag all
user@R1# commit

```

2. Run the show log command.

```

user@R1> show log ipv6-nd-trace
Mar 29 14:07:16 trace_on: Tracing to "/var/log/P1/ipv6-nd-trace" started
Mar 29 14:07:16.287229 background dispatch running job ipv6_ra_delete_interface_config_job
for task Router-Advertisement
Mar 29 14:07:16.287452 task_job_delete: delete background job

```

```

ipv6_ra_delete_interface_config_job for task Router-Advertisement
Mar 29 14:07:16.287505 background dispatch completed job ipv6_ra_delete_interface_config_job
for task Router-Advertisement
Mar 29 14:07:16.288288 ipv6_ra_iflchange(Router-Advertisement): ifl 0xb904378 ifl fe-1/2/2.9
104 change 0, intf 0xba140d8
Mar 29 14:07:16.288450 ipv6_ra_iflchange(Router-Advertisement): ifl 0xb904250 ifl fe-1/2/0.1
85 change 0, intf 0xba14000
Mar 29 14:07:16.288656 ipv6_ra_iflchange(Router-Advertisement): ifl 0xb9044a0 ifl fe-1/2/1.5
80 change 0, intf 0xba1406c
Mar 29 14:07:16.289293 ipv6_ra_ifachange(Router-Advertisement): ifa 0xba002bc
fe80::2a0:a514:0:54c ifl fe-1/2/1.5 80 change 0, intf 0xba1406c
Mar 29 14:07:16.289358 -- nochange/add
Mar 29 14:07:16.289624 ipv6_ra_ifachange(Router-Advertisement): ifa 0xba00230
2001:db8:0:5:2a0:a514:0:54c ifl fe-1/2/1.5 80 change 0, intf 0xba1406c
Mar 29 14:07:16.289682 -- nochange/add
Mar 29 14:07:16.289950 ipv6_ra_ifachange(Router-Advertisement): ifa 0xba001a4
fe80::2a0:a514:0:14c ifl fe-1/2/0.1 85 change 0, intf 0xba14000
Mar 29 14:07:16.290009 -- nochange/add
Mar 29 14:07:16.290302 ipv6_ra_ifachange(Router-Advertisement): ifa 0xba00118
2001:db8:0:1:2a0:a514:0:14c ifl fe-1/2/0.1 85 change 0, intf 0xba14000
Mar 29 14:07:16.290365 -- nochange/add
Mar 29 14:07:16.290634 ipv6_ra_ifachange(Router-Advertisement): ifa 0xba003d4
fe80::2a0:a514:0:94c ifl fe-1/2/2.9 104 change 0, intf 0xba140d8
Mar 29 14:07:16.290694 -- nochange/add
Mar 29 14:07:16.290958 ipv6_ra_ifachange(Router-Advertisement): ifa 0xba00348
2001:db8:0:9:2a0:a514:0:94c ifl fe-1/2/2.9 104 change 0, intf 0xba140d8
Mar 29 14:07:16.291017 -- nochange/add
Mar 29 14:07:20.808516 task_job_create_foreground: create job ipv6 ra for task Router-
Advertisement
Mar 29 14:07:20.808921 foreground dispatch running job ipv6 ra for task Router-Advertisement
Mar 29 14:07:20.809027 ipv6_ra_send_advertisement: sending advertisement for ifl 104 to
ff02::1
Mar 29 14:07:20.809087 (4810916) sending advertisement for ifl 104
Mar 29 14:07:20.809170 ifa 0xba00348 2001:db8:0:9:2a0:a514:0:94c/64
Mar 29 14:07:20.809539 --> sent 56 bytes
Mar 29 14:07:20.809660 task_timer_reset: reset Router-Advertisement_ipv6ra
Mar 29 14:07:20.809725 task_timer_set_oneshot_latest: timer Router-Advertisement_ipv6ra
interval set to 7:07
Mar 29 14:07:20.809772 foreground dispatch completed job ipv6 ra for task Router-Advertisement

```

## RELATED DOCUMENTATION

[Supported IPv4, TCP, and UDP Standards](#)[Supported IPv6 Standards](#)[Accessing Standards Documents on the Internet](#)

# Secure IPv6 Neighbor Discovery

## SUMMARY

The Secure Neighbor Discovery (SEND) Protocol for IPv6 traffic prevents an attacker who has access to the broadcast segment from abusing NDP or ARP to trick hosts into sending the attacker traffic destined for someone else, a technique known as ARP poisoning.

## IN THIS SECTION

- [Understanding Secure IPv6 Neighbor Discovery | 19](#)
- [Example: Configuring Secure IPv6 Neighbor Discovery | 20](#)

## Understanding Secure IPv6 Neighbor Discovery

One of the functions of the IPv6 Neighbor Discovery Protocol (NDP) is to resolve network layer (IP) addresses to link layer (for example, Ethernet) addresses, a function performed in IPv4 by Address Resolution Protocol (ARP). The Secure Neighbor Discovery (SEND) Protocol prevents an attacker who has access to the broadcast segment from abusing NDP or ARP to trick hosts into sending the attacker traffic destined for someone else, a technique known as ARP poisoning.

To protect against ARP poisoning and other attacks against NDP functions, SEND should be deployed where preventing access to the broadcast segment might not be possible.

SEND uses RSA key pairs to produce cryptographically generated addresses, as defined in RFC 3972, *Cryptographically Generated Addresses (CGA)*. This ensures that the claimed source of an NDP message is the owner of the claimed address.



## Example: Configuring Secure IPv6 Neighbor Discovery

### IN THIS SECTION

- [Requirements | 20](#)
- [Overview | 20](#)
- [Configuration | 22](#)
- [Verification | 24](#)

This example shows how to configure IPv6 Secure Neighbor Discovery (SEND).

### Requirements

This example has the following requirements:

- Junos OS Release 9.3 or later
- IPv6 deployed in your network
- If you have not already done so, you must generate or install an RSA key pair.

To generate a new RSA key pair, enter the following command:

```
user@host> request security pki generate-key-pair type rsa certificate-id certificate-id-name  
size size
```

### Overview

#### IN THIS SECTION

- [Topology | 21](#)

To configure SEND, include the following statements:

```
protocols {
  neighbor-discovery {
    onlink-subnet-only;
    secure {
      security-level {
        (default | secure-messages-only);
      }
      cryptographic-address {
        key-length number;
        key-pair pathname;
      }
      timestamp {
        clock-drift number;
        known-peer-window seconds;
        new-peer-window seconds;
      }
      traceoptions {
        file filename <files number> <match regular-expression> <size size> <world-
readable | no-world-readable>;
        flag flag;
        no-remote-trace;
      }
    }
  }
}
```

Specify **default** to send and receive both secure and unsecured Neighbor Discovery Protocol (NDP) packets. To configure SEND to accept secured NDP messages only and to drop unsecured ones, specify **secure-messages-only**.

All nodes on the segment need to be configured with SEND if the **secure-messages-only** option is used, which is recommended unless only a small subset of devices require increased protection. Failure to configure SEND for all nodes might result in loss of connectivity.

## Topology

## Configuration

### IN THIS SECTION

- Procedure | 22

### 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.

```
set protocols neighbor-discovery secure security-level secure-messages-only
set protocols neighbor-discovery secure cryptographic-address key-length 1024
set protocols neighbor-discovery secure cryptographic-address key-pair /var/etc/rsa_key
set protocols neighbor-discovery secure timestamp
```

#### 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 [CLI User Guide](#).

To configure a secure IPv6 neighbor discovery:

1. Configure the security level.

```
[edit protocols neighbor-discovery secure]
user@host# set security-level secure-messages-only
```

2. (Optional) Enable the key length.

The default key length is 1024.

```
[edit protocols neighbor-discovery secure]
user@host# set cryptographic-address key-length 1024
```

3. (Optional) Specify the directory path of the public-private key file generated for the cryptographic address.

The default location of the file is the `/var/etc/rsa_key` directory.

```
[edit protocols neighbor-discovery secure]
user@host# set cryptographic-address key-pair /var/etc/rsa_key
```

4. (Optional) Configure a timestamp to ensure that solicitation and redirect messages are not being replayed.

```
[edit protocols neighbor-discovery secure]
user@host# set timestamp
```

## Results

From configuration mode, confirm your configuration by entering the `show protocols` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
user@host# show protocols
neighbor-discovery {
  secure {
    security-level {
      secure-messages-only;
    }
    cryptographic-address {
      key-length 1024;
      key-pair /var/etc/rsa_key;
    }
    timestamp;
  }
}
```

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

## Verification

### IN THIS SECTION

- [Checking the IPv6 Neighbor Cache | 24](#)
- [Tracing Neighbor Discovery Events | 24](#)

Confirm that the configuration is working properly.

### Checking the IPv6 Neighbor Cache

#### Purpose

Display information about the IPv6 neighbors.

#### Action

From operational mode, enter the `show ipv6 neighbors` command.

#### Meaning

In IPv6, the Address Resolution Protocol (ARP) has been replaced by the NDP. The IPv4 command `show arp` is replaced by the IPv6 command `show ipv6 neighbors`. The key pieces of information displayed by this command are the IP address, the MAC (Link Layer) address, and the interface.

### Tracing Neighbor Discovery Events

#### Purpose

Perform additional validation by tracing SEND.

## Action

1. Configure trace operations.

```
[edit protocols neighbor-discovery secure]
user@host# set traceoptions file send-log
user@host# set traceoptions flag all
```

2. Run the show log command.

```
user@host> show log send-log
Apr 11 06:21:26 proto: outgoing pkt on idx 68 does not have CGA (fe80::2a0:a514:0:14c),
dropping pkt
Apr 11 06:26:44 proto: sendd_msg_handler: recv outgoing 96 bytes on idx 70 with offset 40
Apr 11 06:26:44 dbg: sendd_proto_handler: Modifier (16)
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Apr 11 06:26:44 cga: snd_is_lcl_cga: BEFORE overriding cc, cc:0, ws->col:0
Apr 11 06:26:44 proto: outgoing pkt on idx 70 does not have CGA (fe80::2a0:a514:0:24c),
dropping pkt
Apr 11 06:26:47 proto: sendd_msg_handler: recv outgoing 96 bytes on idx 68 with offset 40
Apr 11 06:26:47 dbg: sendd_proto_handler: Modifier (16)
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

## Meaning

The output shows that because the packet does not have a cryptographically generated address, the packet is dropped.

# NDP Proxy and DAD Proxy

## SUMMARY

This topic provides details on Neighbor Discovery Protocol (NDP) proxy and Duplicate Address Detection (DAD) proxy functionality for interface restricted and interface unrestricted mode.

## IN THIS SECTION

- [Overview | 26](#)
- [Configuring NDP Proxy | 28](#)
- [Configuring DAD Proxy | 29](#)
- [NDP Proxy Support For User Routes | 30](#)

## Overview

### IN THIS SECTION

- [NDP and DAD Proxy \(Interface Restricted Mode\) | 27](#)
- [NDP and DAD Proxy \(Interface Unrestricted Mode\) | 27](#)

The NDP proxy functionality enables packet forwarding among the hosts that are in the same subnet and restricted from communicating directly with each other. NDP proxy is required when you want to enable a host device on different physical segments with same subnet to communicate without an additional gateway and prefix. NDP proxy is like node or a router in the middle of multiple segments with same prefix.

When you configure the device as NDP proxy for addresses, the configured proxy interface (proxy router or node) sends the Neighbor Advertisement (NA) replies to Neighbor Solicitation (NS) on behalf of devices in a different physical segment.

The DAD proxy functionality enables a device to respond to DAD queries for a node that cannot communicate directly with other nodes in the same subnet.



**NOTE:** NDP or DAD proxy functionality does not work if the NS is for a link local address.

## NDP and DAD Proxy (Interface Restricted Mode)

The NDP proxy functionality (interface restricted mode) enables packet forwarding among the hosts that are in the same subnet and restricted to communicate directly with each other. This functionality is primarily used in a scenario where the proxy node needs to apply access control and intercept traffic flowing among the hosts. When you configure NDP proxy on an SRX Series Firewall, the device sends NA and responds to requests from devices seeking MAC addresses of IPv6 prefixes assigned to hosts inside the SRX Series Firewall.

The DAD feature detects the usage of duplicate addresses on a local link by using Neighbor Solicitation (NS) messages. The DAD feature is for IPv6 address and functions similar to gratuitous ARP in IPv4.

## NDP and DAD Proxy (Interface Unrestricted Mode)

Starting in Junos OS Release 22.1R1, we support NDP and DAD proxy functionality across multiple proxy configured interfaces (interface unrestricted mode). NDP interface unrestricted proxy works within the existing IPv6 ND functionality and is invoked only if its enabled. Interface unrestricted mode the ND functionality works together across all the configured interfaces for NDP and DAD proxy.

In earlier releases, NDP and DAD proxy functionality was limited and restricted to only the configured interface. Currently, NDP and DAD proxy functionality works across the multiple configured interfaces (interface unrestricted mode).

With NDP and DAD proxy functionality in interface unrestricted mode, the configured interfaces function together to send Neighbor Advertisement (NA) replies to Neighbor Solicitation (NS) on behalf of nodes in a different physical segment which are not directly reachable by the nodes in the originating segment without the overhead of additional prefix assignment.

When you enable NDP proxy in interface unrestricted mode on interfaces using the `set interfaces interface-name unit number family inet6 ndp-proxy interface-unrestricted` command, the proxy interfaces:

- Generates NA for NS requests. Requests are then sent from hosts on behalf of other hosts that are reachable on the subnet through the proxy interfaces.
- Generates NS and sends to all proxy interfaces for the subnet, when the requested address in NS is not available in the neighbor table.

Looks for forwardable routes for the targeted address in the route table that belongs to the ingress interface of the NS packet. Route lookup provides list of routes pointing to resolve next hops. Proxy uses these next hops to send NS on different ports configured.



**NOTE:** When the proxy does the route lookup and the resulting route next hop points to the same interface where the NS has arrived, then proxy drops that NS.



- Allows you to enforce Neighbor Unreachability Detection (NUD) even if the requested target address is available in neighbor cache and is reachable. The force ND feature is useful when the hosts move from one segment to another. To enable the NDP proxy force resolve functionality use the set protocols neighbor-discovery ndp-proxy proxy-force-resolve command.
- Forwards packets between hosts that it proxies, allowing communication between the hosts, once the neighbors are resolved.

The DAD feature detects the usage of duplicate addresses on a local link by using Neighbor Solicitation (NS) messages.

When you enable DAD proxy on multiple interfaces using the set interfaces *interface-name* unit *<number>* family inet6 dad-proxy interface-unrestricted command:

- DAD proxy generates NA reply for the DAD NS requests on behalf of other hosts, if the NS tentative address is reachable through other proxy interface.
- When a DAD NS request arrives and if the tentative address is not available or in stale state in the neighbor cache, the DAD proxy initiates NUD on all other proxy interfaces except the received one.
- If a DAD request is from a host for a tentative address that is already in the middle of a DAD process by another host, then DAD proxy replies with NA for both hosts.

## Configuring NDP Proxy

You can enable Neighbor Discovery Protocol (NDP) proxy in interface restricted mode or interface unrestricted mode (across multiple interfaces). You cannot configure both DAD proxy interface restricted mode and interface unrestricted mode simultaneously on an interface.

1. To enable NDP proxy restricted to an interface (interface restricted mode):

```
set interfaces interface-name family inet6 ndp-proxy interface-restricted
```

2. To enable NDP proxy on multiple interfaces (interface unrestricted mode):

```
set interfaces interface-name unit number family inet6 ndp-proxy interface-unrestricted
```

3. To enable or disable NDP proxy behavior of sending NS for already learnt entries that are reachable:

```
set protocols neighbor-discovery ndp-proxy proxy-force-resolve
```

4. To disable NDP proxy for an address that is not present in neighbor cache:

```
set protocols neighbor-discovery ndp-proxy no-proxy-on-resolve
```

5. To get the statistics of events such as NDP proxy requests, NDP proxy conflicts, NDP proxy duplicates, NDP proxy resolve requests and dropped NDP packets:

```
show system statistics icmp6
```

## Configuring DAD Proxy

You can enable Duplicate Address Detection (DAD) proxy on a restricted interface (interface restricted mode) or across multiple interfaces (interface unrestricted mode). You cannot configure DAD proxy in interface restricted mode and interface unrestricted modes simultaneously.

To configure DAD proxy on an interface or on multiple interfaces:

1. To enable DAD proxy restricted to an interface (interface restricted mode):

```
set interfaces interface-name family inet6 dad-proxy interface-restricted
```

2. To enable DAD proxy on multiple interfaces (interface unrestricted mode):

```
set interfaces interface-name unit <number> family inet6 dad-proxy interface-unrestricted
```

3. To disable DAD proxy for an address that is not present in a neighbor cache:

```
set protocols neighbor-discovery dad-proxy no-proxy-on-resolve
```

4. To get the statistics of events such as DAD proxy requests, DAD proxy conflicts, DAD proxy duplicates, DAD proxy resolve requests and dropped DAD packets:

```
show system statistics icmp6
```

## NDP Proxy Support For User Routes

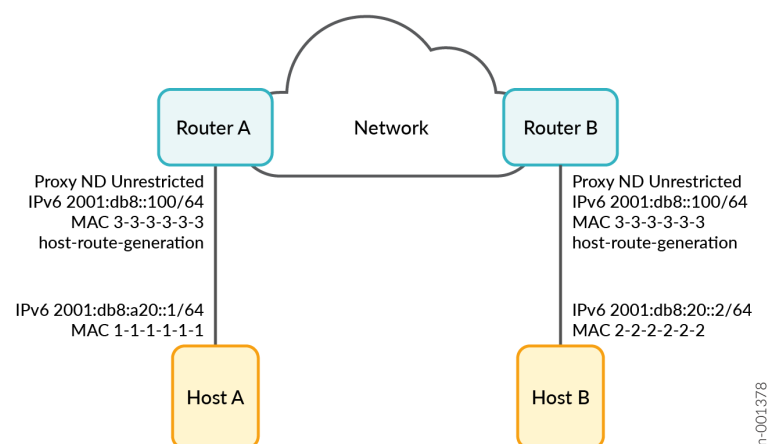
### SUMMARY

This topic introduces Neighbor Discovery Protocol (NDP) Proxy support. When this feature is enabled for an interface, host routes are maintained in the Routing Information Base (RIB) for each IPv4 or IPv6 host address present in the Address Resolution Protocol (ARP) and Neighbor Discovery Protocol (NDP) tables.

Let's take for example, two hosts that are part of the same network segment but are located on different physical networks, each connected to its own gateway. Both the hosts are configured with the same IP address.

When servers are divided into hosts to support flexible deployment and migration across servers and gateways, it is common to configure Layer 2 connectivity between multiple gateways. However, this can result in network issues, such as larger Layer 2 domains and broadcast storms. To resolve this, you can use a route-based proxy Neighbor Discovery on the host gateways. The gateway sends its own MAC address to the source host and the traffic sent from the source host to other hosts is done over routes, instead of Layer 2 switching.

Figure 2: Proxy ND topology



Let's consider Host A and Host B with IPV6 addresses 2001:db8:a20::1/64 and 2001:db8:20::2/64 to be on the same network segment. Router A and Router B are connected using a Layer 3 network. The

interface to the hosts on both routers has the same MAC and IPv6 address. As the destination IPv6 address and local IPv6 address are on the same network segment, if Host A wants to communicate with Host B, it sends a Neighbor Solicitation (NS) packet to request for Host B's MAC address. However, since Host A and Host B are on different physical networks, Host B doesn't receive the NS and cannot reply.

To address the problem, a new proxy Neighbor Discovery (ND) option based on user routes is introduced. This mode is supported only on unrestricted mode of proxy.

The new proxy mode relies on the host route generation feature to check reachability. When this feature is enabled for an interface, host routes are maintained in the Routing Information Base (RIB) for each IPv4 or IPv6 host address present in the Address Resolution Protocol (ARP) and Neighbor Discovery Protocol (NDP) tables. These host routes that are active in the RIB are available for redistribution into BGP or IGP, subject to routing protocol export policies. The user route proxy ND relies on these redistributed user routes for checking reachability of hosts.



**NOTE:** Proactive ARP detection helps quickly learn reachable VMs and Proactive IPv6 neighbor learning is not supported.



**NOTE:** Host routes in non-default routing tables are not supported and therefore, the new user route based proxying is also not supported.

## SEE ALSO

*user-route-proxy*

*show system statistics icmp6*

# Neighbor Discovery Cache Protection

## SUMMARY

NDP Cache Protection enables you to protect the routing engine from certain types of denial-of-service (DoS) attacks in IPv6 deployment scenarios.

## IN THIS SECTION

● [Neighbor Discovery Cache Protection Overview | 32](#)

- [Configuring Neighbor Discovery Cache Protection | 33](#)
- [Example: Configuring Neighbor Discovery Cache Protection to Prevent Denial-of-Service Attacks | 34](#)

## Neighbor Discovery Cache Protection Overview

Routing Engines can be susceptible to certain denial-of-service (DoS) attacks in IPv6 deployment scenarios. IPv6 subnets in general tend to be very large—for example, a /64 subnet might have a high number of unassigned addresses. The control plane of the Routing Engine performs the address resolution for unknown addresses. An attacker can quickly overwhelm the control plane of the Routing Engine by generating resolution requests for this unassigned address space, resulting in a cache overflow. The attacker relies on both the number of requests generated and the rate at which requests are queued up. Such scenarios can tie up router resources and prevent the Routing Engine from answering valid neighbor solicitations and maintaining existing neighbor cache entries, effectively resulting in a DoS attack for legitimate users.

The strategies for mitigating such DoS attacks are as follows:

- Filter unused address space.
- Minimize the size of subnets.
- Configure discard routes for subnets.
- Enforce limits to the size and rate of resolution for entries in the neighbor discovery cache.

Neighbor discovery cache impact can be minimized by restricting the number of IPv6 neighbors and new unresolved next-hop addresses that can be added to the cache. You can set limits per interface by using the `nd6-max-cache` and the `nd6-new-hold-limit` configuration statements or system-wide by using the `nd-system-cache-limit` configuration statement.



### NOTE:

- For small sized platforms such as ACX, EX22XX, EX3200, EX33XX, and SRX, default is 20,000.

- For medium sized platforms such as EX4200, EX45XX, EX4300, EX62XX, QFX, and MX, default is 75,000.
- For rest of the platforms, default is 100,000.

## Configuring Neighbor Discovery Cache Protection

Routing Engines can be susceptible to certain types of denial-of-service (DoS) attacks in IPv6 deployment scenarios. IPv6 subnets in general tend to be very large; for example, a /64 subnet might have a high number of unassigned addresses. The control plane of the Routing Engine performs the address resolution for unknown addresses. An attacker can quickly overwhelm the control plane of the Routing Engine by generating resolution requests for this unassigned address space, resulting in a cache overflow. An attacker relies on both the number of requests generated and the rate at which requests are queued up.

The neighbor discovery process is that part of the control plane that implements the Neighbor Discovery Protocol. It is responsible for performing address resolution and maintaining the entries in the neighbor cache. One way to mitigate the DoS attacks is by enforcing limits to the size of the neighbor discovery cache and the rate of resolution of new next-hop entries, and by prioritizing certain categories of neighbor discovery traffic. You can configure limits to the neighbor discovery cache per interface and systemwide.

Before you begin, ensure that you are running Junos OS Release 15.1 or later.

Local limits apply to individual interfaces and are defined for resolved and unresolved entries in the neighbor discovery queue, while global limits apply systemwide.

To configure neighbor discovery cache protection on an interface:

1. Configure IPv6 family for the interface.

```
[edit interfaces interface-name unit unit number family]
user@host# set inet6
```

2. Configure the maximum size of the neighbor discovery cache for the interface.

```
[edit interfaces interface-name unit unit number family inet6]
user@host# set nd6-max-cache limit
```

3. Configure the maximum number of unresolved entries in the neighbor discovery cache that can be attached to the interface.

```
[edit interfaces interface-name unit unit number family inet6]
user@host# set nd6-new-hold-limit limit
```

To verify the configuration, execute the `show interfaces interface-name operational` command.

To configure neighbor discovery cache protection systemwide:

- Configure the systemwide limit for the neighbor discovery cache.

```
[edit]
user@host# set system nd-system-cache-limit limit
```

To verify the configured system-wide limits, execute the `show system statistics icmp6 operational` command.



#### NOTE:

- For small sized platforms such as ACX, EX22XX, EX3200, EX33XX, and SRX, default is 20,000.
- For medium sized platforms such as EX4200, EX45XX, EX4300, EX62XX, QFX, and MX, default is 75,000.
- For rest of the platforms, default is 100,000.

## Example: Configuring Neighbor Discovery Cache Protection to Prevent Denial-of-Service Attacks

### IN THIS SECTION

- [Requirements | 35](#)
- [Overview | 35](#)
- [Configuration | 36](#)

This example shows how to configure a limit to the number of IPv6 neighbor entries that can be added to the neighbor discovery. Enforcing limits to the number of entries in the cache mitigates denial-of-service (DoS) attacks. The neighbor discovery cache feature supports two types of limits:

- **Local**—Local limits are configured per interface and are defined for resolved and unresolved entries in the neighbor discovery cache.
- **Global**—Global limits apply systemwide. A global limit is further defined separately for the public interfaces and management interfaces, for example, fxp0. The management interface has a single global limit and no local limit. The global limit enforces a systemwide cap on entries for the neighbor discovery cache, including for the loopback interface for the internal routing instance, as well as management interfaces and the public interfaces.

## Requirements

This example requires MX Series routers running Junos OS Release 15.1 or later.

## Overview

Routing Engines can be susceptible to certain types of DoS attacks in IPv6 deployment scenarios. IPv6 subnets in general tend to be very large—for example, a /64 subnet might have a high number of unassigned addresses, which can be used to perform DoS attacks. The control plane of the Routing Engine performs the address resolution for unknown addresses. An attacker can quickly overwhelm the control plane of the Routing Engine by generating resolution requests for this unassigned address space and overflow the queue. The attacker relies on both the number of requests generated and the rate at which requests are queued up.

The neighbor discovery process is that part of the control plane that implements the Neighbor Discovery Protocol. It is responsible for performing address resolution and maintaining the neighbor cache. One way to mitigate DoS attacks is by enforcing limits on the neighbor discovery queue limits, which can be done by restricting queue size and the rate of resolution, and by prioritizing certain categories of neighbor discovery traffic.



## Configuration

### IN THIS SECTION

- [CLI Quick Configuration | 36](#)
- [Configuring Neighbor Discovery Cache Protection | 37](#)
- [Results | 37](#)

To configure neighbor discovery cache protection, perform these tasks:

### 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 copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` from configuration mode.

```
set interfaces ge-0/3/0 unit 5 family inet6 nd6-max-cache 100
set interfaces ge-0/3/0 unit 5 family inet6 nd6-new-hold-limit 100
```

You can also configure a systemwide limit to the number of IPv6 neighbor entries in the neighbor discovery cache. This limit also includes the loopback interface, management interfaces, and the public interfaces.

```
set system nd-system-cache-limit 100
```

The limit distribution from the `nd-system-cache-limit` statement for different interface types is performed according to certain fixed percentages. When `nd-system-cache-limit` is defined as  $X$  and the internal routing interface neighbor discovery cache limit is  $Y$  (default is 200), then:

- Public maximum cache limit,  $Z = 80\%$  of  $(X - Y)$
- Management interface maximum cache limit (for example, `fxp0`),  $M = 20\%$  of  $(X - Y)$

## Configuring Neighbor Discovery Cache Protection

### 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 [CLI User Guide](#).

To configure neighbor discovery cache protection per interface:

- Configure the `nd6-max-cache` and `nd6-new-hold-limit`.

```
[edit]
user@host# set interfaces ge-0/3/0 unit 5 family inet6 nd6-max-cache 100
user@host# set interfaces ge-0/3/0 unit 5 family inet6 nd6-new-hold-limit 100
```

### Results

To confirm neighbor discovery cache protection locally, enter `show interfaces ge-0/3/0` from configuration mode. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces ge-0/3/0
unit 5{
  family inet6 {
    nd6-max-cache 100;
    nd6-new-hold-limit 100;
  }
}
```

## Verification

### IN THIS SECTION

- [Verifying Neighbor Discovery Cache Protection Globally | 38](#)
- [Verifying Neighbor Discovery Cache Protection Locally | 39](#)

Confirm that the configuration is working properly.

## Verifying Neighbor Discovery Cache Protection Globally

### Purpose

Verify that the output reflects the systemwide limit for the neighbor discovery cache.

### Action

From operational mode, run the `show system statistics icmp6` command.

```
user@host> show system statistics icmp6

icmp6:
    79 Calls to icmp_error
    0 Errors not generated because old message was icmp error
    0 Errors not generated because rate limitation
    Output histogram:
        79 unreachable
        30 echo
        163 multicast listener query
        6 multicast listener report
        940 neighbor solicitation
        694184 neighbor advertisement
    0 Messages with bad code fields
    0 Messages < minimum length
    0 Bad checksums
    0 Messages with bad length
    Input histogram:
        10 echo reply
        6 multicast listener report
        693975 neighbor solicitation
    Histogram of error messages to be generated:
        0 No route
        0 Administratively prohibited
        0 Beyond scope
        79 Address unreachable
        0 Port unreachable
        0 Time exceed transit
        0 Time exceed reassembly
        0 Erroneous header field
```

```

0 Unrecognized next header
0 Unrecognized option
0 Unknown
0 Message responses generated
0 Messages with too many ND options
100000 Max System ND nh cache limit
79840 Max Public ND nh cache limit
200 Max IRI ND nh cache limit
19960 Max Management intf ND nh cache limit
79840 Current Public ND nexthops present
4 Current IRI ND nexthops present
0 Current Management ND nexthops present
909266 Total ND nexthops creation failed as limit reached
909266 Public ND nexthops creation failed as public limit reached
0 IRI ND nexthops creation failed as iri limit reached
0 Management ND nexthops creation failed as mgt limit reached

```

## Meaning

The systemwide cap enforced on the neighbor discovery cache entries is **100000**.

**Management ND nexthops creation failed as mgt limit reached** indicates the drop count for the management interface when the systemwide limit is reached. **Total ND nexthops creation failed as limit reached** indicates failure for management, public, or Internal routing instance interfaces, and **Public ND nexthops creation failed as public limit reached** indicates the drop count for public interfaces when the systemwide limit to the number of entries is reached.

## Verifying Neighbor Discovery Cache Protection Locally

### Purpose

Verify that the output reflects the configured interface limits.

### Action

From operational mode, run the `show interfaces ge-0/3/0` command.

```

user@host> show interfaces ge-0/3/0
Logical interface ge-0/2/0.8 (Index 348) (SNMP ifIndex 690)
  Flags: Up SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.8 ] Encapsulation: ENET2
  Input packets : 181628

```

```

Output packets: 79872
Protocol inet6, MTU: 1500
Max nh cache: 100000, New hold nh limit: 100000, Curr nh cnt: 79840, Curr new hold cnt: 0,
NH drop cnt: 0
  Flags: Is-Primary
  Addresses, Flags: Is-Preferred Is-Primary
    Destination: 8001:1::/64, Local: 8001:1::1:1
  Addresses, Flags: Is-Preferred
    Destination: fe80::/64, Local: fe80::56e0:3200:8c6:e0a4
Protocol multiservice, MTU: Unlimited

```

## Meaning

The maximum number of total entries and the maximum number of entries for new unresolved next-hop addresses that can be attached to interface ge-0/3/0 is **100000**.

**NH drop cnt** refers to the number of neighbor discovery requests not serviced because the interface maximum queue size limits have been reached.

## RELATED DOCUMENTATION

[IPv6 Neighbor Discovery Overview | 2](#)

*nd-system-cache-limit*

*nd6-max-cache*

*nd6-new-hold-limit*

# Router Advertisement Proxy

## SUMMARY

Starting in Junos OS Release 22.1R1, we support Router Advertisement (RA) proxy functionality on SRX Series Firewalls and vSRX Virtual Firewall 3.0. With this functionality, the device can proxy the RA

## IN THIS SECTION

- [Overview | 41](#)
- [Configure RA Proxy | 42](#)

packets from service provider router to the clients (host).

## Overview

### IN THIS SECTION

- [Benefits](#) | 42

An IPv6 network deployment usually has one or more upstream routers to delegate IPv6 prefixes through Router Advertisement (RA) to clients. When a client connects to the network, the client starts sending Router Solicitations (RS) IPv6 requests. When clients send, upstream routers either respond with unicast (Layer 2 or Layer 3) RA or with multicast RA. Whenever a new client joins the network, a unicast or a multicast RA is sent to from the router to the client. If it is a multicast packet, then the existing clients also receive the RA, which results in traffic increase. The solution for handling the increased traffic is to enable IPv6 RA proxy to monitor the incoming unsolicited RA and RS packets.

RA proxy functionality conveys all the information that is received from the routers to the clients. The RA proxy information includes the following:

- Router Preference
- Router lifetime
- Reachable time
- Retransmit timer
- ICMPv6 Option (Source link-layer address)
- ICMPv6 Option (MTU)
- ICMPv6 Option (Prefix information)
- ICMPv6 Option (Route Information)
- ICMPv6 Option (DNS Search List)
- ICMPv6 Option (RDNSS option)



**NOTE:** The RA is processed as well as proxied, unless proxying is disabled. Also, when RA proxy is enabled, the RA packets received on the upstream interface proxied to all the downstream interfaces. The RA packets received on the downstream interface are not proxied to all the upstream interfaces.

## Benefits

- Helps in management of traffic by snooping incoming unsolicited RA and Router Solicitations packets allowing transmission of information from service provider side routers to the clients.
- Loops are prevented using RA blackout timer.
- New proxy bit provides an indication that the RA packet is proxied.

## Configure RA Proxy

To enable RA proxy on an interface:

1. Configure the interface as upstream (service provider side interface) for RA proxy:

```
set protocols router-advertisement interface <interface-name> upstream-mode
```

2. Configure the interface as downstream (host side interface) for RA proxy:

```
set protocols router-advertisement interface <interface-name> downstream-mode
```

3. Configure the list of downstream interfaces for RA proxy:

```
set protocols router-advertisement interface <interface-name> downstream <downstream-interface-name>
```

4. Configure preference to select configured or proxied parameters for downstream interface.

```
set protocols router-advertisement interface <interface-name> parameter-preference <preference ((configured | proxied)>
```

5. Configure passive mode option on an interface. If passive mode is configured on the interface, the interface receives and processes RA packets. The interface does not send RAs in (receive-only mode). The commit fails if the interface has both downstream and passive-mode option configured simultaneously. To enable passive mode (RA receive only mode) on an interface:

```
set protocols router-advertisement interface <interface-name> passive-mode
```

To view the details of configured RA proxy options listed below, use the `show ipv6 router-advertisement`.

- Upstream interfaces
- Downstream interfaces
- Proxy flag
- Proxy blackout timer
- Passive mode details

## SEE ALSO

---

[\*downstream\*](#)

---

[\*downstream-mode\*](#)

---

[\*upstream-mode\*](#)

---

[\*parameter-preference\*](#)

---

[\*passive-mode\*](#)



# 2

CHAPTER

## Troubleshooting

---

### IN THIS CHAPTER

- [Troubleshooting Network Issues | 45](#)
-

# Troubleshooting Network Issues

## IN THIS SECTION

- [Working with Problems on Your Network | 45](#)
- [Isolating a Broken Network Connection | 46](#)
- [Identifying the Symptoms of a Broken Network Connection | 48](#)
- [Isolating the Causes of a Network Problem | 50](#)
- [Taking Appropriate Action for Resolving the Network Problem | 51](#)
- [Evaluating the Solution to Check Whether the Network Problem Is Resolved | 53](#)
- [Checklist for Tracking Error Conditions | 55](#)
- [Configure Routing Protocol Process Tracing | 57](#)
- [Configure Routing Protocol Tracing for a Specific Routing Protocol | 60](#)
- [Monitor Trace File Messages Written in Near-Real Time | 63](#)
- [Stop Trace File Monitoring | 64](#)

## Working with Problems on Your Network

### IN THIS SECTION

- [Problem | 45](#)
- [Solution | 46](#)

### 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

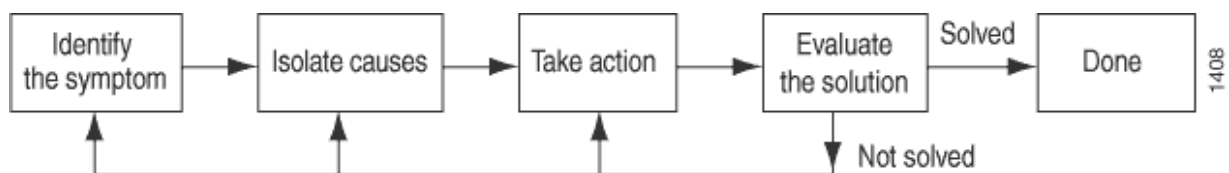
**Table 1: Checklist for Working with Problems on Your Network**

Tasks	Command or Action
<i>Isolating a Broken Network Connection</i>	
1. <i>Identifying the Symptoms of a Broken Network Connection</i>	<b>ping (ip-address   hostname) show route (ip-address   hostname) traceroute (ip-address   hostname)</b>
1. <i>Isolating the Causes of a Network Problem</i>	show < configuration   interfaces   protocols   route >
1. <i>Taking Appropriate Action for Resolving the Network Problem</i>	[edit] delete routing options static route destination-prefix commit and-quit show route destination-prefix
1. <i>Evaluating the Solution to Check Whether the Network Problem Is Resolved</i>	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 3 on page 46](#), 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.

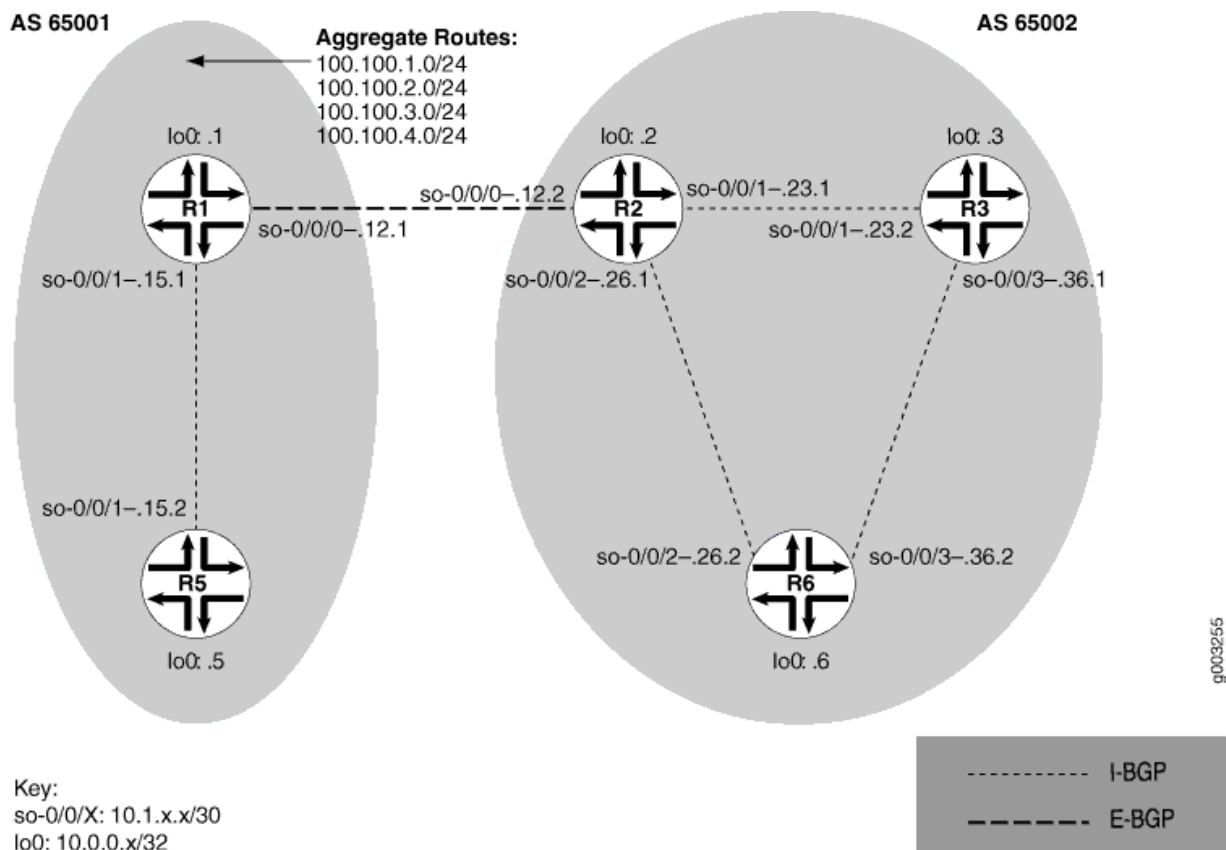
**Figure 3: Process for Diagnosing Problems in Your Network**



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 4 on page 47 shows the network topology used in this topic to illustrate the process of diagnosing problems in a network.

Figure 4: Network with a Problem



The network in Figure 4 on page 47 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 | 48
- Solution | 48

### 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 | 50
- Solution | 50

### 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 | 52
- Solution | 52



## 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 | 53
- Solution | 54

### 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 | 55
- Solution | 55

## Problem

### Description

Table 2 on page 55 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 2: Checklist for Tracking Error Conditions

Tasks	Command or Action
Configure Routing Protocol Process Tracing	
1. <i>Configure Routing Protocol Process Tracing</i>	[edit] edit routing-options traceoptions <i>filename</i> size <i>size</i> files <i>number</i> show con log <i>filename</i>
1. <i>Configure Routing Protocol Tracing for a Specific Routing Protocol</i>	[edit] edit protocol <i>protocol-name</i> trace <i>filename</i> size <i>size</i> files <i>number</i> show con log <i>filename</i>
1. <i>Monitor Trace File Messages Written in Near-Real Time</i>	monitor start <i>filename</i>
1. <i>Stop Trace File Monitoring</i>	monitor stop <i>filename</i>

Table 2: 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 send detail show commit run show log <i>filename</i>
1. Display Sent or Received BGP Packets	[edit] edit protocol bgp traceoptions send (send   receive) show commit run show log
1. Diagnose BGP Session Establishment Problems	[edit] edit protocol bgp set traceoptions send detail show commit run show log <i>filename</i>
Configure IS-IS-Specific Options	
1. Displaying Detailed IS-IS Protocol Information	[edit] edit protocol isis traceoptions send detail show commit run show log <i>filename</i>
1. Displaying Sent or Received IS-IS Protocol Packets	[edit] edit protocols isis traceoptions send (send   receive) show commit run show log
1. Analyzing IS-IS Link-State PDUs in Detail	[edit] edit protocols isis traceoptions send detail show commit run show log <i>filename</i>
Configure OSPF-Specific Options	
1. Diagnose OSPF Session Establishment Problems	[edit] edit protocols ospf traceoptions send detail show commit run show log <i>filename</i>
1. Analyze OSPF Link-State Advertisement Packets in Detail	[edit] edit protocols ospf traceoptions send update detail show commit run show log

## Configure Routing Protocol Process Tracing

### IN THIS SECTION

- [Action | 57](#)
- [Meaning | 59](#)

### 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 3 on page 59 lists tracing flags and example output for Junos-supported routing protocol daemon tracing.

**Table 3: Routing Protocol Daemon Tracing Flags**

Tracing Flag	Description	Example Output
<b>all</b>	All operations	Not available.
<b>general</b>	Normal operations and routing table change	Not available.
<b>normal</b>	Normal operations	Not available.
<b>policy</b>	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
<b>route</b>	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
<b>state</b>	State transitions	Not available.



Table 3: 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 | 60
- Meaning | 62

### 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 4 on page 62](#) 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 4: Standard Trace Options for Routing Protocols**

Tracing Flag	Description
<b>all</b>	All operations

Table 4: Standard Trace Options for Routing Protocols *(Continued)*

Tracing Flag	Description
<b>general</b>	Normal operations and routing table changes
<b>normal</b>	Normal operations
<b>policy</b>	Policy operations and actions
<b>route</b>	Routing table changes
<b>state</b>	State transitions
<b>task</b>	Interface transactions and processing
<b>timer</b>	Timer usage

## Monitor Trace File Messages Written in Near-Real Time

### IN THIS SECTION

- Purpose | 63
- Action | 64

### 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 | 65](#)
- [Sample Output | 65](#)

## 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>
```

# 3

CHAPTER

## Configuration Statements and Operational Commands

---

### IN THIS CHAPTER

- [Junos CLI Reference Overview](#) | 67
-

# 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](#)