Nokia Service Router Linux

Release 23.10

Interfaces Guide

3HE 19846 AAAA TQZZA
Edition: 01
November 2023

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

Table of contents

1 About this guide

This document describes interfaces used with the Nokia Service Router Linux (SR Linux). Examples of commonly used commands are provided.

This document is intended for network technicians, administrators, operators, service providers, and others who need to understand how the router is configured.

Note:
This manual covers the current release and may also contain some content that will be released in later maintenance loads. See the *SR Linux Release Notes* for information on features supported in each load.

Configuration and command outputs shown in this guide are examples only; actual displays may differ depending on supported functionality and user configuration.

1.1 Precautionary and information messages

The following are information symbols used in the documentation.

DANGER: Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.

WARNING: Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.

Caution: Caution indicates that the described activity or situation may reduce your component or system performance.

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

1.2 Conventions

Nokia SR Linux documentation uses the following command conventions.

- **Bold** type indicates a command that the user must enter.
- Input and output examples are displayed in `Courier` text.
- An open right-angle bracket indicates a progression of menu choices or simple command sequence (often selected from a user interface). Example: **start** > **connect to**.
- A vertical bar (|) indicates a mutually exclusive argument.

- Square brackets ([ ]) indicate optional elements.
- Braces ({ }) indicate a required choice. When braces are contained within square brackets, they indicate a required choice within an optional element.
- *Italic* type indicates a variable.

Generic IP addresses are used in examples. Replace these with the appropriate IP addresses used in the system.

2 What's new

Topic	Location
Support for IPv4 unnumbered interfaces	IPv4 unnumbered interfaces
IPv4 ARP and IPv6 ND information is now added, including support for an IPv6 neighbor limit on subinterfaces.	IPv4 ARP and IPv6 ND Configuring IPv6 neighbor limit on subinterfaces

3 Interfaces

On the SR Linux, an interface is any physical or logical port through which packets can be sent to or received from other devices. The SR Linux supports the following interface types:

- Loopback

  A loopback interface is a virtual interface that is always up, providing a stable source or destination from which packets can always be originated or received. The SR Linux supports up to 256 loopback interfaces system-wide, across all network-instances. Loopback interfaces are named `lo`*N*, where *N* is 0 to 255.

- System

  The system interface is a type of loopback interface that has characteristics that do not apply to regular loopback interfaces:

  – The system interface can be bound to the default network-instance only.

  – The system interface does not support multiple IPv4 addresses or multiple IPv6 addresses.

  – The system interface cannot be administratively disabled. Once configured, it is always up.

  The SR Linux supports a single system interface named `system0`. When the system interface is bound to the default network-instance, and an IPv4 address is configured for it, the IPv4 address is the default local address for multi-hop BGP sessions to IPv4 neighbors established by the default network-instance, and it is the default IPv4 source address for IPv4 VXLAN tunnels established by the default network-instance. The same functionality applies with respect to IPv6 addresses / IPv6 BGP neighbors / IPv6 VXLAN tunnels.

- Network

  Network interfaces carry transit traffic, as well as originate and terminate control plane traffic and in-band management traffic.

  The physical ports in line cards installed in the SR Linux are network interfaces. A typical line card has a number of front-panel cages, each accepting a pluggable transceiver. Each transceiver may support a single channel or multiple channels, supporting one Ethernet port or multiple Ethernet ports, depending on the transceiver type and its breakout options.

  In the SR Linux CLI, each network interface has a name that indicates its type and its location in the chassis. The location is specified with a combination of slot number and port number, using the following formats:

  `ethernet-`*`slot/port`*

  For example, interface `ethernet-2/1` refers to the line card in slot 2 of the SR Linux chassis, and port 1 on that line card.

- Management

  Management interfaces are used for out-of-band management traffic. The SR Linux supports a single management interface named `mgmt0`.

  The `mgmt0` interface supports the same functionality and defaults as a network interface, except for the following:

  – Packets sent and received on the `mgmt0` interface are processed completely in software.

  - The `mgmt0` interface does not support multiple output queues, so there is no output traffic differentiation based on forwarding class.
  - The `mgmt0` interface does not support pluggable optics. It is a fixed 10/100/1000-BaseT copper port.

- Integrated routing and bridging (IRB)

  IRB interfaces enable inter-subnet forwarding. Network instances of type **mac-vrf** are associated with a network instance of type **ip-vrf** via an IRB interface.

  IRB interfaces are named `irb`*N*, where *N* is 0 to 255. See IRB interfaces.

On the SR Linux, each loopback, network, management, and IRB interface can be subdivided into one or more subinterfaces. See Subinterfaces.

4 Linux interface naming conventions

Every type of SR Linux interface has an underlying interface in the Linux OS. These interfaces have names that adhere to Linux restrictions (maximum 15 characters and no slashes). The Linux interface name formats are as follows:

- Loopback interfaces: `lo`*`N`*, where *N* is 0 to 255; for example, `lo0`
- Network interfaces: `e`*`slot-port-subinterface`*; for example, `e4-2-1`
- Management interface: `mgmt0`
- System interface: `system0`
- LAG interface: `lag`*`N`*
- IRB interface: `irb`*`N`*

5 Basic interface configuration

The following example shows a configuration for interface basic parameters, including administratively enabling the interface, specifying a description, and setting the MTU. The settings apply to any subinterfaces on the port, unless overridden in the subinterface configuration.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/2
 interface ethernet-1/2
    description Sample_interface_config
    admin-state enable
    mtu 1500
```

6 Subinterfaces

On the SR Linux, each loopback, network, management, and IRB interface can be subdivided into one or more subinterfaces. A subinterface is a logical channel within its parent interface.

Traffic belonging to one subinterface can be distinguished from traffic belonging to other subinterfaces of the same port using encapsulation methods such as 802.1Q VLAN tags.

While each port can be considered a shared resource of the router that is usable by all network-instances, a subinterface can only be associated with one network-instance at a time. To move a subinterface from one network-instance to another, you must disassociate it from the first network-instance before associating it with the second network-instance. See the *SR Linux Interfaces Guide*.

You can configure ACL policies to filter IPv4 and IPv6 packets entering or leaving a subinterface. See the *SR Linux ACL and Policy-Based Routing Guide.*

The SR Linux supports policies for assigning traffic on a subinterface to forwarding classes or remarking traffic at egress before it leaves the router. DSCP classifier policies map incoming packets to the appropriate forwarding classes, and DSCP rewrite-rule policies mark outgoing packets with an appropriate DSCP value based on the forwarding class.

6.1 Routed and bridged subinterfaces

SR Linux subinterfaces can be specified as type routed or bridged:

- Routed subinterfaces can be assigned to a network-instance of type **mgmt**, **default**, or **ip-vrf**.
- Bridged subinterfaces can be assigned to a network-instance of type **mac-vrf**.

Routed subinterfaces allow for configuration of IPv4 and IPv6 settings, and bridged subinterfaces allow for configuration of bridge table and VLAN ingress/egress mapping.

6.2 Subinterface naming conventions

The CLI name of a subinterface is the name of its parent interface followed by a dot (`.`) and an index number that is unique within the scope of the parent interface. For example, the subinterface named `ethernet-2/1.0` is a subinterface of `ethernet-2/1`, and it has index number `0`.

- Each loopback interface (`lo`*`N`*) can only have one subinterface, and the index number can be in the range 0 to 255.
- Each network interface (`ethernet-`*`slot/port`*) where the **vlan-tagging** parameter is set to **false** can have one subinterface, and the index number can be in the range 0 to 9999.
- Each network interface where the **vlan-tagging** parameter is set to **true** can have up to 4096 subinterfaces (up ro 1024 of type routed and 3072 of type bridged) with each subinterface assigned a unique index number in the range 0 to 9999.
- The management and system interfaces (`mgmt0` and `system0`) can only have one subinterface, with an index number of 0.

The Linux name of a subinterface adheres to Linux restrictions (maximum 15 characters and no slashes). For example, the subinterface named `ethernet-2/1.0` has the Linux name `e2-1.0`.

6.3 Basic subinterface configuration

For IPv4 packets to be sourced from a subinterface, the IPv4 address family must be enabled on the subinterface and the subinterface must be configured with an IPv4 address and prefix length that indicates the other IPv4 hosts reachable on the same subnet.

A subinterface can have up to 64 IPv4 prefixes assigned to it. One or more of these can be optionally configured as a primary candidate. Within the set of IPv4 prefixes configured as primary candidates, the lowest IPv4 address that does not fail duplicate address detection is selected as the primary address for the subinterface. The primary address is used by upper layer protocols that need to choose only one IPv4 address from which to source their messages, as well as for information about this interface displayed with the **info from state** command. If there is no suitable address in the set of IPv4 prefixes configured as primary candidates (or if no IPv4 prefix is configured as primary), a selection is made from the IPv4 prefixes not configured as primary candidates.

For IPv6 packets to be sourced from a subinterface, the IPv6 address family must be enabled on the subinterface, which must be configured with a global unicast IPv6 address and prefix length. The address can be configured statically or obtained from a DHCP server.

A subinterface can have up to 16 global unicast IPv6 addresses and prefixes assigned to it. One or more of these can be optionally configured as a primary candidate. Within the set of IPv6 prefixes configured as primary candidates, the lowest IPv6 address that does not fail duplicate address detection is selected as the primary address for the subinterface. The primary address is used by upper layer protocols that need to choose only one IPv6 address from which to source their messages, as well as for information about this interface displayed with the **info from state** command. If there is no suitable address in the set of IPv6 prefixes configured as primary candidates (or if no IPv6 prefix is configured as primary), a selection is made from the IPv6 prefixes not configured as primary candidates.

The following example shows basic parameters for a subinterface configuration, including IPv4 and IPv6 addresses and prefix lengths.

The configuration for subinterface 1 administratively enables the subinterface, specifies an ACL policy for input IPv4 traffic, and specifies a DSCP classifier policy that assigns input IPv4 traffic to a queue based on the 6-bit DSCP value in the IP header.

The configuration for subinterface 2 administratively enables the subinterface, and configures multiple IPv4 and IPv6 addresses and prefix lengths. The primary IPv4 address for the subinterface is selected from among the set of IPv4 prefixes configured as primary candidates; the selected IPv4 address is the numerically lowest address that does not fail duplicate address detection. The global unicast IPv6 address for the subinterface is selected from the IPv6 prefix configured as primary. The selected global unicast IPv6 address is the numerically lowest address that does not fail duplicate address detection.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/2
    interface ethernet-1/2
        description Sample_interface_config
        admin-state enable
        mtu 1500
        subinterface 1 {
            admin-state enable
            ipv4 {
                admin-state enable
```

```
                dhcp-client true {
                }
            }
            ipv6 {
                admin-state enable
                address 2001:1::192:168:12:1/126 {
                }
            }
            acl {
                input {
                    ipv4-filter 101
                }
            }
            qos {
                input {
                    classifiers {
                        ipv4-dscp 1
                    }
                }
            }
        }
        subinterface 2 {
            admin-state enable
            ipv4 {
                admin-state enable
                address 192.168.12.1/30 {
                    primary
                }
                address 192.168.12.2/30 {
                    primary
                }
                address 192.168.12.2/30 {
                }
            }
            ipv6 {
                admin-state enable
                address 2001:1::192:168:12:2/126 {
                    primary
                }
                address 2001:1::192:168:12:3/126 {
                }
            }
        }
```

6.4 IPv4 unnumbered interfaces

An IPv4 unnumbered interface is an interface of the router that is enabled for sending and receiving IPv4 packets, but has no configured IPv4 address of its own. When the router needs to send IPv4 packets from this interface, it borrows the IPv4 source address from another interface that is numbered; that is, one that has its own, explicitly configured IPv4 address. The IPv4 address can only be borrowed from an interface that is up, so the borrowed address is almost always a loopback or system address.

IPv6 unnumbered interfaces exist, but they are not widely used because an IPv6-enabled interface always has an IPv6 link-local address. The only benefit of borrowing an IPv6 address from another interface is to borrow a global-unicast IPv6 address from the other interface. SR Linux does not support IPv6 unnumbered interfaces.

Using IPv4 unnumbered interfaces can lead to simplified network design, where each router has minimally just one IPv4 address, assigned to a loopback/system interface. When Router A is connected by a link to

Router B, Router A borrows the address of its loopback, and Router B borrows the address of its loopback. There is no need to allocate a unique IPv4 subnet for the link and no need to coordinate the addresses at each end. This allows a plug-and-play approach and greatly simplifies the reconfiguration effort required if physical connectivity is changed.

Unnumbered interfaces can also simplify control plane protection configuration because CPM-filters (or the equivalent) protect traffic that is terminated by the loopback/system address and require no modification when new interfaces are added.

Routing protocol support for unnumbered interfaces on SR Linux is as follows:

- OSPFv2 and OSPFv3 adjacencies can form over unnumbered interfaces, but they are always P2P (point-to-point) adjacencies. If the interface-type is configured as broadcast, the operational interface-type is forced back to point-to-point.
- IS-IS adjacencies can form over unnumbered interfaces, but they are always P2P adjacencies. If the interface-type is configured as broadcast, the operational interface-type is forced back to point-to-point.
- iBGP and multi-hop eBGP sessions can form over unnumbered interfaces.
- LDP supports unnumbered interfaces.

IPv4 unnumbered interfaces are supported on routed subinterfaces of `ethernet` and `lag`*`N`* interfaces.

A subinterface of a network-instance can borrow an IPv4 address from any other subinterface of that network-instance, as long as that other subinterface is not a `mgmt` subinterface.

If the interface configuration points to a subinterface in a different network-instance or a subinterface not bound to any network-instance at all, the subinterface remains operationally down with respect to IPv4 forwarding. If the interface configuration points to a subinterface with multiple IPv4 addresses, the primary IPv4 address of the subinterface is the borrowed address.

If the interface configuration points to a subinterface that is operationally down (or where IPv4 is operationally down), the borrowing subinterface remains operationally down with respect to IPv4 forwarding. If the interface configuration points to a subinterface that has no IPv4 addresses, the borrowing subinterface remains operationally down with respect to IPv4 forwarding.

When an ICMP message is transmitted from an unnumbered IPv4 subinterface, the source address is the borrowed address.

6.4.1 Configuring an IPv4 unnumbered subinterface

Procedure

To configure an IPv4 unnumbered subinterface, you administratively enable the unnumbered subinterface and specify the interface the unnumbered subinterface borrows its address from. If you configure a subinterface as unnumbered, you cannot assign an IPv4 address to it.

Example

The following example enables an IPv4 unnumbered subinterface. The subinterface is configured to borrow the address of loopback interface lo0.1.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/8 subinterface 1 ipv4
    interface ethernet-1/8 {
        subinterface 1 {
            ipv4 {
                admin-state enable
```

```
                unnumbered {
                    admin-state enable
                    interface lo0.1
                }
            }
        }
    }
```

6.5 IPv6 address assignment on SR Linux

A routed subinterface becomes operational for IPv6 forwarding when `interface.subinterface.ipv6.admin-state` is set to `enable`.

To assign a global-unicast IPv6 address to a routed subinterface, you must provide a host address and a prefix-length. In SR Linux, the address and prefix-length are subject to the following restrictions:

- `::/128` is disallowed (special reserved address for unspecified)
- `::1/128` is disallowed (special reserved address for loopback)
- Addresses in the block `ff00::/8` are disallowed (special reserved addresses for multicast)

In SR Linux, some types of routed subinterfaces support assignment of multiple IPv6 global-unicast addresses to a single interface. The maximum number of IPv6 global unicast addresses that can be assigned to each routed subinterface is summarized in the following table.

Table 1: Maximum IPv6 global-unicast addresses for routed subinterface types

Subinterface	Maximum IPv6 addresses
system0.0	1
mgmt0.0	1
lo<N>.x	16
lag<N>.x	16
irb<N>.x	16
ethernet-<slot>/<connector>/<port>.x	16

On a particular routed subinterface, each global-unicast IPv6 address must be unique, but the subnets that result from applying the prefix-length masks can create overlap. For example, it is possible to assign `2001::1/64`, `2001::2/64`, and `2001::3/64` to the same subinterface. This can lead to ambiguity when a locally originated packet is destined for an address such as `2001::5`. SR Linux chooses the source address based on the longest prefix match of the destination address; if there are still multiple choices, the numerically lowest IP address is selected. For this example, `2001::1` is used as a local address when you issue a **ping 2001::5** command.

On a routed subinterface that supports multiple global-unicast IPv6 addresses, one of them is selected as the primary IPv6 address. The primary address is used by upper-layer protocols that need to choose only one IPv6 global-unicast address from which to source their messages. You can influence the election algorithm by configuring one or more IPv6 addresses as primary; this set of IPv6 addresses are the preferred candidates. The system elects the lowest IP address among the members of the preferred set that did not fail duplicate address detection (DAD). If no addresses are preferred or else all the preferred

addresses fail DAD validation then the system elects the lowest IP address among the members of the non-preferred set that did not fail duplicate address detection (DAD). The elected address is the single address that has the `primary` leaf in the output of `info from state` commands.

IPv6 LLA assignment

If a subinterface has a statically configured IPv6 address in the IPv6 link-local block (`FE80::/10`), the system publishes this address as the IPv6 link-local address (LLA) of the interface. If the configuration of the subinterface has no IPv6 address in the IPv6 link-local block, the system generates an IPv6 link-local address for assignment to the subinterface.

The link-local IPv6 address is generated from the MAC address of the subinterface, which is itself derived from the MAC address of the port. The address generation process converts the 48-bit MAC address into a 64-bit EUI-64 address, and this EUI-64 address becomes the host portion of the `FE80::/10` prefix.

When the static or generated IPv6 LLA is programmed, the subinterface can receive IPv6 packets with any of the following destinations:

- IPv6 LLA
- Solicited-node multicast address for the LLA
- `ff02::1` (all IPv6 devices)
- `ff02::2` (all IPv6 routers)

To terminate (as a host) IPv6 unicast packets originated by remote devices (that is, devices not connected to the local link), one or more IPv6 global unicast addresses must be assigned to the subinterface.

Several routing protocols including BFD, IS-IS, OSPFv3, and BGP can establish a session between two interfaces on the same link that have a link-local address but no IPv6 global-unicast address.

- For BGP, the local-address of a static BGP session of any type cannot be configured as an LLA; however, the local TCP connection endpoint (and BGP next-hop-self) automatically uses the LLA when the neighbor address is link-local.

  If the neighbor address is defined as an LLA (with the subinterface name to scope the address), the session comes up only if it is eBGP. iBGP peers do not come up because they are presumed to be multi-hop.

  Dynamic sessions are not accepted if they come from an IPv6 LLA.
- BFD sessions tied to IS-IS adjacencies that were discovered from a Hello that included an IPv6 LLA, and BFD sessions tied to BGP sessions established between link-local addresses do not establish.

6.5.1 Assigning IPv6 addresses to a subinterface

Procedure

You can configure one or more IPv6 addresses on a subinterface. This table lists the number of global-unicast IPv6 addresses that can be configured on each type of subinterface.

Example: Configuring primary IPv6 addresses

The following example configures three IPv6 addresses on a subinterface. Two addresses are configured as primary. Upper-layer protocols prefer the primary addresses when selecting an IPv6 address from which to source their messages.

```
--{ * candidate shared default }--[  ]--
```

```
# info interface ethernet-1/12 subinterface 1 ipv6
    interface ethernet-1/12 {
        subinterface 1 {
            ipv6 {
                admin-state enable
                address 2001:db8:1:1::2/64 {
                    primary
                }
                address 2001:db8:2:1::2/64 {
                    primary
                }
                address 2001:db8:3:1::2/64 {
                }
            }
        }
    }
```

Example: Configuring a static IPv6 link-local address

The following example configures a static IPv6 LLA for a subinterface.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/12 subinterface 1 ipv6
    interface ethernet-1/12 {
        subinterface 1 {
            ipv6 {
                admin-state enable
                address fe80::1:2/64 {
                    type link-local-unicast
                }
            }
        }
    }
```

The system automatically generates an IPv6 LLA (`fe80::/10` + EUI-64 identifier derived from the MAC address) if the `ipv6` container is present in the configuration. When you configure a static IPv6 LLA, as in the example above, the configured static LLA overrides the system-generated LLA. If you delete the static LLA, the subinterface reverts to the system-generated LLA. Only one LLA is supported per subinterface.

6.6 Subinterface VLAN configuration

When the **vlan-tagging** parameter is set to **true** for a network interface, the interface can accept ethertype 0x8100 frames with one or more VLAN tags. The interface can be configured with up to 4096 subinterfaces, each with a separate index number.

The following example enables VLAN tagging for an interface and configures two subinterfaces. Single-tagged packets received on subinterface `ethernet-2/1.1` are encapsulated with VLAN ID 101.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-2/1
    interface ethernet-2/1 {
        admin-state enable
        vlan-tagging true
        subinterface 1 {
            admin-state enable
            vlan {
                encap {
```

```
                    single-tagged {
                        vlan-id 100
                    }
                }
            }
        }
        subinterface 2 {
            admin-state enable
            vlan {
                encap {
                    single-tagged {
                        vlan-id 101
                    }
                }
            }
        }
    }
```

6.7 VLAN tag TPID configuration

The 802.1Q VLAN Tag Protocol Identifier (TPID) in the VLAN tag of an Ethernet frame indicates the protocol type of the VLAN tag. This feature allows you to configure the VLAN tag TPID that is used to classify frames as Dot1q on single-tagged interfaces or to push at egress; by default, the value of the VLAN tag TPID is 0x8100.

You can configure the following TPID values for an interface:

- `TPID_0X8100`
  Default value typically used to identify 802.1Q single-tagged frames.
- `TPID_0X88A8`
  Typical TPID value for 802.1Q provider bridging or QinQ S-tags.
- `TPID_0X9100`
  Alternate TPID value for QinQ tags.
- `TPID_0X9200`
  Alternate TPID value for QinQ tags.
- `TPID_ANY`
  Wildcard that matches any of the generally used TPID values for single- or multi-tagged VLANs. This value is equivalent to matching any of `TPID_0X8100`, `TPID_0X88A8`, `TPID_0X9100`, or `TPID_0x9200` at ingress. At egress, if a tag needs to be pushed and `TPID_ANY` is configured, the default TPID value is used.

Note:

- This feature does not change the behavior of subinterfaces of type `untagged` or `any`, nor does it change the behavior of interfaces configured with `vlan-tagging false`.
- On a Dot1q interface, if the configured TPID is (for example) `TPID_0X88A8`, the service-delimiting tags have TPID value 0x88a8, so frames received with that TPID may match a subinterface if they come with the appropriate VLAN ID. Frames with any other TPID value only match untagged interfaces or tagged interfaces with VLAN ID `any` or `untagged`.
- Only one TPID value can be configured per interface.
- The TPID pushed at egress is one of the following:

  - The configured TPID, if SR Linux is pushing a service-delimiting tag and the configured TPID is different from TPID_ANY.
  - The default TPID of 0x8100, if SR Linux is pushing a service-delimiting tag and the configured TPID is TPID_0X8100 or TPID_ANY.
- This feature is supported on all interfaces that support VLAN tagging (that is, all interfaces except for loopback, system, management, and IRB).
- For CPU injected packets, the configured interface TPID is used in injected unicast and multicast frames (in the context of the MAC-VRF flood group), so the configured TPID appears in CPM-outgoing Ethernet frames.
- When the TPID is configured on a LAG interface, the configuration is propagated to all LAG members.
- This feature is not supported on interfaces configured in breakout mode.

6.7.1 Configuring the VLAN tag TPID for an interface

Procedure

To configure the VLAN tag TPID for an interface, specify one of the TPID values listed in the previous topic.

Example

The following example configures the TPID_ANY wildcard for an interface. At ingress, this configuration matches TPID_0X8100, TPID_0X88A8, TPID_0X9100, or TPID_0x9200. SR Linux pushes the default TPID of 0x8100 to egress frames.

```
--{ candidate shared default }--[  ]--
# info interface ethernet-1/1
    interface ethernet-1/1 {
        vlan-tagging true
        tpid TPID_ANY
        subinterface 1 {
            vlan {
                encap {
                    single-tagged {
                        vlan-id 101
                    }
                }
            }
        }
    }
```

6.7.2 Displaying the VLAN tag TPID

Procedure

Use the **show interface detail** command to display the VLAN tag TPID for an interface.

Example

```
--{ candidate shared default }--[  ]--
# show interface ethernet-1/12 detail
==========================================================================
```

```
Interface: ethernet-1/12
--------------------------------------------------------------------------------
  Description     : dut2
  Oper state      : up
  Down reason     : N/A
  Last change     : 47m6s ago, 1 flaps since last clear
  Speed           : 100G
  Flow control    : Rx is disabled, Tx is disabled
  MTU             : 9232
  VLAN tagging    : true
  VLAN TPID       : 0x8100
  Queues          : 8 output queues supported, 1 used since the last clear
  MAC address     : 00:01:02:FF:00:0C
  Last stats clear: 47m6s ago
  Breakout mode   : false
```

6.8 Bridged subinterface configuration

Bridged subinterfaces are associated with a mac-vrf network-instance (see the Network-instances chapter in the *SR Linux Configuration Basics*). On mac-vrf network instances, traffic can be classified based on VLAN tagging. Interfaces where VLAN tagging is set to false or true can be used with mac-vrf network instances.

A default subinterface can be specified, which captures untagged and non-explicitly configured VLAN-tagged frames in tagged subinterfaces.

Within a tagged interface, a default subinterface (**vlan-id** value is set to **any**) and an untagged subinterface can be configured. This kind of configuration behaves as follows:

- The **vlan-id any** subinterface captures untagged and non-explicitly configured VLAN-tagged frames.
- The untagged subinterface captures untagged and packets with tag0 as outermost tag.

When **vlan-id any** and untagged subinterfaces are configured on the same tagged interface, packets for unconfigured VLANs go to the **vlan-id any** subinterface, and tag0/untagged packets go to the untagged subinterface.

The **vlan-id** value can be configured as a specific valid number or with the keyword **any**, which means any frame that does not hit the **vlan-id** configured in other subinterfaces of the same interface is classified in this subinterface.

In the following example, the `vlan encap untagged` setting is enabled for subinterface 1. This setting allows untagged frames to be captured on tagged interfaces.

For subinterface 2, the `vlan encap single-tagged vlan-id any` setting allows non-configured VLAN IDs and untagged traffic to be classified to this subinterface.

With the `vlan encap untagged` setting on one subinterface, and the `vlan encap single-tagged vlan-id any` setting on the other subinterface, traffic enters the appropriate subinterface; that is, traffic for unconfigured VLANs goes to subinterface 2, and tag0/untagged traffic goes to subinterface 1.

```
--{ candidate shared default }--[  ]--
# info interface ethernet-1/2
 interface ethernet-1/2
  vlan-tagging true
  subinterface 1 {
    type bridged
    vlan {
      encap {
```

```
      untagged
           }
       }
subinterface 2 {
  type bridged
  vlan {
    encap {
      single-tagged {
        vlan-id any
    }
```

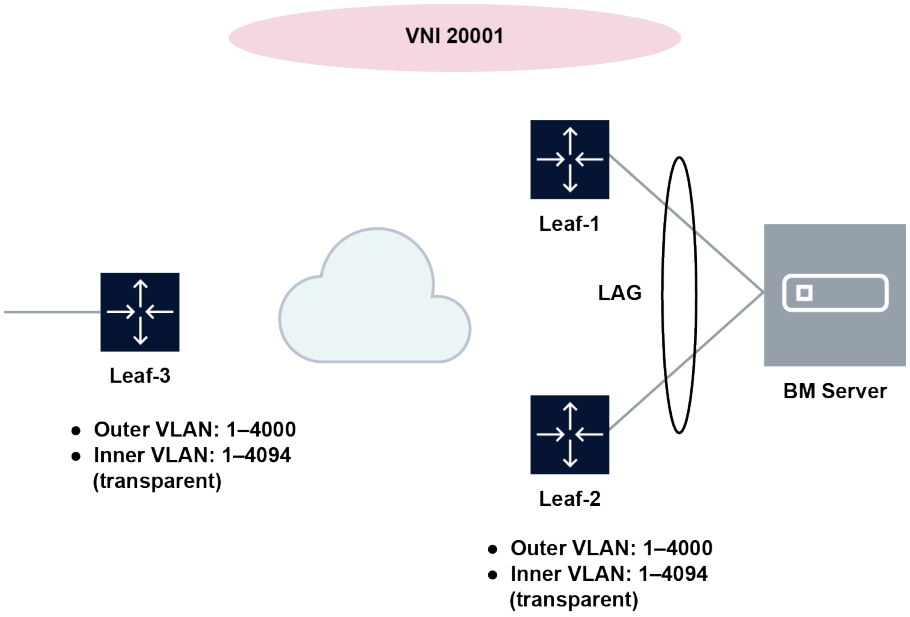
6.9 Dot1q VLAN ranges on bridged subinterfaces

Bridged subinterfaces support Dot1q VLAN ranges. When a bridged subinterface is configured for a Dot1q VLAN range, traffic matching any VLAN in the range is associated with the bridged subinterface.

For example, you can configure a bridged subinterface with the Dot1q VLAN range 10 to 100. When an attached device sends traffic that has a VLAN ID in the range 10 through 100, the traffic is associated with the bridged subinterface.

A bridged subinterface can support a single VLAN range or multiple VLAN ranges. The following figure shows a configuration with a single VLAN range.

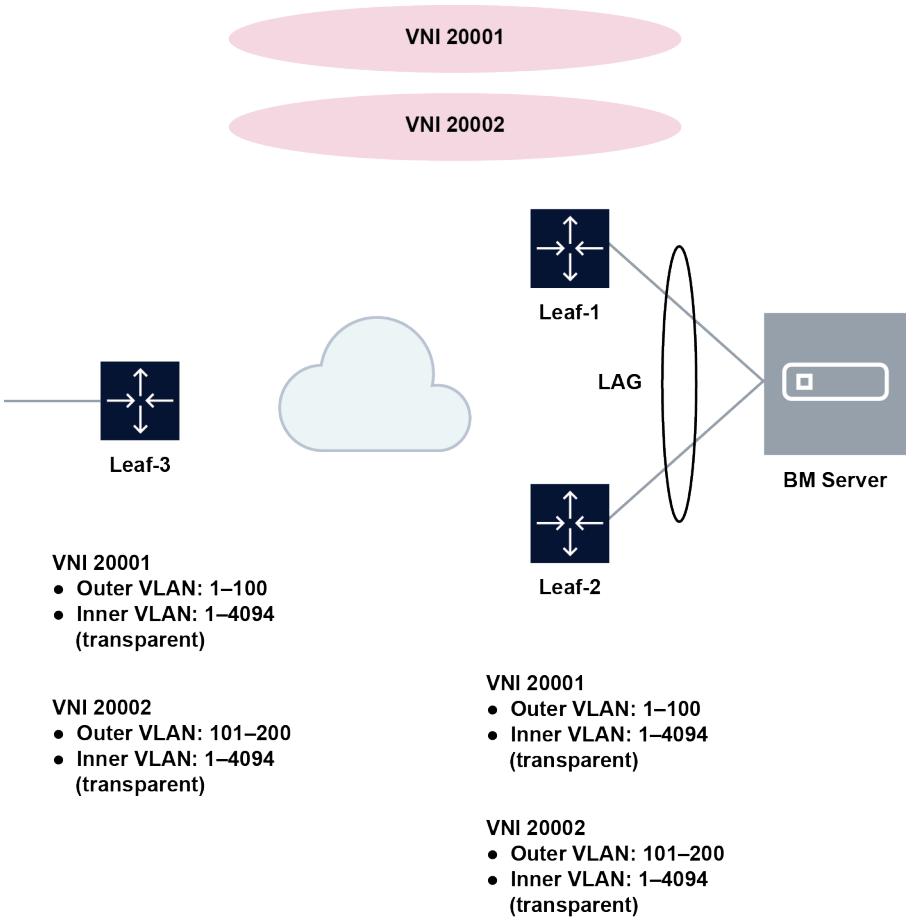
Figure 1: Single VLAN range on a bridged subinterface



In this figure, the Bare Metal (BM) server is attached to a MAC VRF, and LAG-bridged subinterfaces are associated with the Dot1q VLAN range 1-4000. Traffic issued by the BM server with a VLAN ID matching any value in the configured range is associated with the bridged subinterface. VLAN IDs 4001-4095 are not associated with the subinterface.

The following figure shows a configuration with multiple VLAN ranges associated with a bridged subinterface.

Figure 2: Multiple VLAN ranges on a bridged subinterface



In this figure, VLAN ranges are selectively associated with a mac-vrf. VLAN IDs 1–100 are associated with MAC-VRF 20001 (which uses VNI 20001), and VLAN IDs 101–200 are associated with MAC-VRF 20002 (which uses VNI 20002).

In both figures, it is expected that all the leaf nodes attached to the same BD use the same VLAN ranges on all their bridged subinterfaces for the mac-vrf. The Dot1q VLAN tags of the incoming frames are not stripped off at the ingress subinterface. At the egress subinterface (which is configured with the same VLAN range as the ingress subinterface), no additional Dot1q tag is pushed onto the frames.

An SR Linux system can support up to 8,000 subinterfaces that have VLAN ranges, with up to 8 VLAN ranges per interface.

VLAN ranges can be configured on bridged subinterfaces of a mac-vrf with or without IRB subinterfaces. When an IRB subinterface is present in the same mac-vrf where the VLAN range subinterface is configured, incoming frames containing tags with a MAC DA equal to the IRB and associated with a Dot1q VLAN range subinterface are dropped. On egress, traffic coming from the IRB is not tagged if the destination is associated with a VLAN range subinterface.

VLAN ranges cannot overlap within the same subinterface or across subinterfaces of the same physical interface. In addition, ranges configured in subinterface *a* and individual VLAN ID values configured in a single-tagged subinterface *b* cannot overlap if *a* and *b* are defined in the same interface.

6.9.1 Configuring Dot1q VLAN ranges on a bridged subinterface

Procedure

To configure a Dot1q VLAN range, specify the lower and upper values for the range. A subinterface can have multiple ranges associated with it.

Example

The following example configures a range of VLAN IDs associated with a bridged subinterface. Traffic matching any VLAN in the range is associated with the bridged subinterface.

```
--{ candidate shared default }--[  ]--
# info interface ethernet-1/1
    interface ethernet-1/1 {
        vlan-tagging true
        subinterface 1 {
            type bridged
            vlan {
                encap {
                    single-tagged-range {
                        low-vlan-id 10 {
                            high-vlan-id 100
                        }
                    }
                }
            }
        }
    }
```

6.9.2 Displaying Dot1q VLAN range information

Procedure

Check the `Encapsulation` field of the **show interface** command for the configured Dot1q VLAN ranges for the interface.

Example

```
--{ candidate shared default }--[  ]--
# show interface lag1.1
================================================================================
  lag1.1 is up
    Network-instance: MAC-VRF-1
    Encapsulation   : vlan-id 1-10, 20-30
    Type            : bridged
================================================================================
```

7 IRB interfaces

Integrated routing and bridging (IRB) interfaces enable inter-subnet forwarding. Network instances of type **mac-vrf** are associated with a network instance of type **ip-vrf** via an IRB interface.

On SR Linux, IRB interfaces are named `irb`*N*, where *N* is 0 to 255. Up to 4095 subinterfaces can be defined under an IRB interface. An ip-vrf network instance can have multiple IRB subinterfaces, while a mac-vrf network instance can refer to only one IRB subinterface.

IRB subinterfaces are type **routed** and cannot be configured as any other type.

IRB subinterfaces operate in the same way as other routed subinterfaces, including support for the following:

- IPv4 and IPv6 ACLs
- DSCP based QoS (input and output classifiers and rewrite rules)
- Static routes and BGP (IPv4 and IPv6 families)
- IP MTU (with the same range of valid values as Ethernet subinterfaces)
- All settings in the subinterface/ipv4 and subinterface/ipv6 containers. For IPv6, the IRB subinterface also gets an IPv6 link local address
- BFD
- Subinterface statistics

IRB interfaces do not support sFlow or VLAN tagging.

7.1 IRB interface configuration

The following example configures an IRB interface. The IRB interface is operationally up when its admin-state is enabled, and its IRB subinterfaces are operationally up when associated with mac-vrf and ip-vrf network instances. At least one IPv4 or IPv6 address must be configured for the IRB subinterface to be operationally up.

```
--{ candidate shared default }--[  ]--
# info interface irb1
    interface irb1 {
        description IRB_Interface
        admin-state enable
        subinterface 1 {
            admin-state enable
            ipv4 {
                admin-state enable
                address 192.168.1.1/24 {
                }
            }
        }
    }
```

8 Displaying interface statistics

Procedure

To display statistics for a specific interface, use the **info from state** command in candidate or running mode, or the **info** command in state mode.

Example

```
--{ candidate shared default }--[  ]--
# info from state interface ethernet-1/2
    interface ethernet-1/2 {
        admin-state enable
        mtu 9232
        loopback-mode false
        ifindex 49150
        oper-state up
        last-change "an hour ago"
        linecard 1
        forwarding-complex 0
        vlan-tagging false
        tpid TPID_0X8100
        statistics {
            in-packets 513
            in-octets 46399
            in-unicast-packets 486
            in-broadcast-packets 6
            in-multicast-packets 19
            in-discarded-packets 0
            in-error-packets 2
            in-fcs-error-packets 0
            out-packets 524
            out-octets 47380
            out-unicast-packets 498
            out-broadcast-packets 6
            out-multicast-packets 20
            out-discarded-packets 0
            out-error-packets 0
            carrier-transitions 1
        }
        traffic-rate {
            in-bps 0
            out-bps 0
        }
        transceiver {
            tx-laser false
            oper-state down
            oper-down-reason not-present
            ddm-events false
            forward-error-correction disabled
        }
        ethernet {
            dac-link-training false
            lacp-port-priority 32768
            port-speed 10G
            hw-mac-address 00:01:03:FF:00:02
            flow-control {
                receive false
```

```
                transmit false
            }
            statistics {
                in-mac-pause-frames 0
                in-oversize-frames 0
                in-jabber-frames 0
                in-fragment-frames 0
                in-crc-error-frames 0
                out-mac-pause-frames 0
                in-64b-frames 0
                in-65b-to-127b-frames 0
                in-128b-to-255b-frames 0
                in-256b-to-511b-frames 0
                in-512b-to-1023b-frames 0
                in-1024b-to-1518b-frames 0
                in-1519b-or-longer-frames 0
                out-64b-frames 0
                out-65b-to-127b-frames 0
                out-128b-to-255b-frames 0
                out-256b-to-511b-frames 0
                out-512b-to-1023b-frames 0
                out-1024b-to-1518b-frames 0
                out-1519b-or-longer-frames 0
            }
        }
    }
```

8.1 Clearing interface statistics

Procedure

You can clear the statistics counters for a specified interface.

Example: Clear all statistics for an interface

```
# tools interface ethernet-1/1 statistics clear
/interface[name=ethernet-1/2]:
    interface ethernet-1/2 statistics cleared
```

Example: Clear queue statistics for an interface

```
# tools interface ethernet-1/2 statistics queue-statistics clear
```

Example: Clear statistics for a specified queue on an interface

```
# tools interface ethernet-1/2 statistics queue-statistics multicast-queue 0 clear
```

9 Displaying subinterface statistics

Procedure

To display statistics for a specific subinterface, enter the context for the subinterface and use the **info from state** command.

Example

```
--{ candidate shared default }--[  ]--
# info from state interface ethernet-1/2 subinterface 1
    interface ethernet-1/2 {
        subinterface 1 {
            type routed
            admin-state enable
            ip-mtu 1500
            name ethernet-1/2.1
            ifindex 32770
            oper-state up
            last-change "a minute ago"
............linecard 1
............forwarding-complex 0
            ipv4 {
                admin-state enable
                allow-directed-broadcast false
                address 192.168.12.2/30 {
                    origin static
                    primary
                    status preferred
                }
                arp {
                    duplicate-address-detection true
                    timeout 14400
                    learn-unsolicited false
                    proxy-arp false
                    neighbor 192.168.12.1 {
                        link-layer-address 00:01:01:FF:00:01
                        origin dynamic
                        expiration-time "3 hours from now"
                        datapath-programming {
                            status success
                        }
                    }
                }
            }
            ipv6 {
                admin-state enable
                address 2001:1::192:168:12:2/126 {
                    origin static
                    primary
                    status preferred
                }
                address fe80::201:3ff:feff:2/64 {
                    origin link-layer
                    status preferred
                }
                neighbor-discovery {
                    duplicate-address-detection true
```

```
                    reachable-time 30
                    stale-time 14400
                    learn-unsolicited none
                    proxy-nd false
                    neighbor 2001:1::192:168:12:1 {
                        link-layer-address 00:01:01:FF:00:01
                        origin dynamic
                        is-router true
                        current-state stale
                        next-state-time "3 hours from now"
                        datapath-programming {
                            status success
                        }
                    }
                    neighbor fe80::201:1ff:feff:1 {
                        link-layer-address 00:01:01:FF:00:01
                        origin dynamic
                        is-router false
                        current-state stale
                        next-state-time "3 hours from now"
                        datapath-programming {
                            status success
                        }
                    }
                }
                router-advertisement {
                    router-role {
                        admin-state disable
                        current-hop-limit 64
                        managed-configuration-flag false
                        other-configuration-flag false
                        max-advertisement-interval 600
                        min-advertisement-interval 200
                        reachable-time 0
                        retransmit-time 0
                        router-lifetime 1800
                    }
                }
            }
            statistics {
                in-packets 45
                in-octets 5457
                in-discarded-packets 0
                in-forwarded-packets 0
                in-forwarded-octets 0
                in-matched-ra-packets 0
                out-forwarded-packets 0
                out-forwarded-octets 0
                out-originated-packets 43
                out-originated-octets 5357
                out-discarded-packets 0
                out-packets 43
                out-octets 5357
            }
            qos {
                input {
                    classifiers {
                        default-forwarding-class fc0
                        default-drop-probability low
                        dscp-policy default
                    }
                }
                output {
                    rewrite-rules {
```

```
                    dscp-policy default
                }
            }
        }
    }
}
```

9.1 Clearing subinterface statistics

Procedure

You can clear the statistics counters for a specified subinterface.

Example

```
# tools interface ethernet-1/2 subinterface 1 statistics clear
/interface[name=ethernet-1/2]/subinterface[index=1]:
    subinterface ethernet-1/2.1 statistics cleared
```

10 Displaying interface status

Procedure

Use the **show interface** command to display the operational state of configured interfaces.

Example: Display the status of all interfaces and subinterfaces in up state

To display the status of all configured interfaces that have operational state up and their subinterfaces that also have operational state up:

```
--{ running }--[  ]--
# show interface
================================================================================
ethernet-1/10 is up, speed 100G, type 100GBASE-CR4 CA-L
  ethernet-1/10.1 is up
    Encapsulation: null
    IPv4 addr    : 192.35.1.0/31 (static)
    IPv6 addr    : 2001:192:35:1::/127 (static, preferred)
    IPv6 addr    : fe80::22e0:9cff:fe78:e2ea/64 (link-layer, preferred)
--------------------------------------------------------------------------------
ethernet-1/21 is up, speed 100G, type 100GBASE-CR4 CA-L
  ethernet-1/21.1 is up
    Encapsulation: null
    IPv4 addr    : 192.45.1.254/31 (static)
    IPv6 addr    : 2001:192:45:1::fe/127 (static, preferred)
    IPv6 addr    : fe80::22e0:9cff:fe78:e2f5/64 (link-layer, preferred)
--------------------------------------------------------------------------------
ethernet-1/22 is up, speed 100G, type 100GBASE-CR4 CA-L
  ethernet-1/22.1 is up
    Encapsulation: null
    IPv4 addr    : 192.45.3.254/31 (static)
    IPv6 addr    : 2001:192:45:3::fe/127 (static, preferred)
    IPv6 addr    : fe80::22e0:9cff:fe78:e2f6/64 (link-layer, preferred)
--------------------------------------------------------------------------------
ethernet-1/3 is up, speed 100G, type 100GBASE-CR4 CA-L
  ethernet-1/3.1 is up
    Encapsulation: null
    IPv4 addr    : 192.57.1.1/31 (static)
    IPv6 addr    : 2001:192:57:1::1/127 (static, preferred)
    IPv6 addr    : fe80::22e0:9cff:fe78:e2e3/64 (link-layer, preferred)
--------------------------------------------------------------------------------
...
================================================================================
Summary
  3 loopback interfaces configured
  8 ethernet interfaces are up
  1 management interfaces are up
  12 subinterfaces are up
================================================================================
```

Example: Display summary information about all interfaces

To display summary information about interfaces that have operational state up or down:

```
--{ running }--[  ]--
# show interface brief
```

```
+---------------+-------------+------------+-------+--------------+
|     Port      | Admin State | Oper State | Speed |     Type     |
+===============+=============+============+=======+==============+
| ethernet-1/1  | enable      | up         | 100G  | 100GBASE-SR4 |
| ethernet-1/2  | enable      | up         |       | 100GBASE-SR4 |
| ethernet-1/3  | disable     | down       |       |              |
| ethernet-1/4  | disable     | down       |       |              |
| ethernet-1/5  | disable     | down       |       |              |
| ethernet-1/6  | disable     | down       |       |              |
| ethernet-1/7  | disable     | down       |       |              |
+---------------+-------------+------------+-------+--------------+
```

Example: Display summary information about a specific interface

To display summary information about a specific interface:

```
--{ running }--[  ]--
# show interface ethernet-1/1 brief
+---------------+-------------+------------+-------+--------------+
|     Port      | Admin State | Oper State | Speed |     Type     |
+===============+=============+============+=======+==============+
| ethernet-1/1  | enable      | up         | 100G  | 100GBASE-SR4 |
+---------------+-------------+------------+-------+--------------+
```

Example: Display summary information about all interfaces and subinterfaces

To display summary information about interfaces and subinterfaces that have operational state `up` or `down`:

```
--{ running }--[  ]--
# show interface all
================================================================================
ethernet-1/1 is down, reason port-admin-disabled
--------------------------------------------------------------------------------
ethernet-1/10 is up, speed 100G, type 100GBASE-CR4 CA-L
  ethernet-1/10.1 is up
    Encapsulation: null
    IPv4 addr    : 192.35.1.0/31 (static)
    IPv6 addr    : 2001:192:35:1::/127 (static, preferred)
    IPv6 addr    : fe80::22e0:9cff:fe78:e2ea/64 (link-layer, preferred)
--------------------------------------------------------------------------------
ethernet-1/11 is down, reason port-admin-disabled
--------------------------------------------------------------------------------
ethernet-1/12 is down, reason port-admin-disabled
--------------------------------------------------------------------------------
...
================================================================================
Summary
  3 loopback interfaces configured
  8 ethernet interfaces are up
  1 management interfaces are up
  12 subinterfaces are up
================================================================================
```

Example: Display summary information about a specific interface and its subinterfaces

To display summary information about a specific interface and its subinterfaces:

```
--{ running }--[  ]--
# show interface ethernet-1/21
================================================================================
```

```
ethernet-1/21 is up, speed 100G, type 100GBASE-CR4 CA-L
  ethernet-1/21.1 is up
    Encapsulation: null
    IPv4 addr    : 192.45.1.254/31 (static)
    IPv6 addr    : 2001:192:45:1::fe/127 (static, preferred)
    IPv6 addr    : fe80::22e0:9cff:fe78:e2f5/64 (link-layer, preferred)
================================================================================
```

Example: Display details about an interface and its subinterfaces

To display details about a specific interface and its subinterfaces:

```
--{ running }--[  ]--
# show interface ethernet-1/3 detail
================================================================================
Interface: ethernet-1/3
--------------------------------------------------------------------------------
  Description     : rifa-difa-1
  Oper state      : up
  Down reason     : N/A
  Last change     : 23m14s ago, No flaps since last clear
  Speed           : 100G
  Flow control    : Rx is disabled, Tx is not supported
  MTU             : 9232
  VLAN tagging    : false
  Queues          : 8 output queues supported, 3 used since the last clear
  MAC address     : 20:E0:9C:78:E2:E3
  Last stats clear: never
--------------------------------------------------------------------------------
Queue Parameter for ethernet-1/3
--------------------------------------------------------------------------------
  Queue-id   Scheduling   Weight
--------------------------------------------------------------------------------
Traffic statistics for ethernet-1/3
--------------------------------------------------------------------------------
       counter         Rx      Tx
  Octets              14241   11724
  Unicast packets     0       0
  Broadcast packets   0       0
  Multicast packets   52      56
  Errored packets     0       0
  FCS error packets   0       N/A
  MAC pause frames    0       N/A
  Oversize frames     0       N/A
  Jabber frames       0       N/A
  Fragment frames     0       N/A
  CRC errors          0       N/A
--------------------------------------------------------------------------------
Traffic rate statistics for ethernet-1/3
--------------------------------------------------------------------------------
    units     Rx   Tx
  kbps rate
--------------------------------------------------------------------------------
Frame length statistics for ethernet-1/3
--------------------------------------------------------------------------------
  Frame length(Octets)   Rx   Tx
  64 bytes               0    0
  65-127 bytes           5    8
  128-255 bytes          0    48
  256-511 bytes          47   0
  512-1023 bytes         0    0
  1024-1518 bytes        0    0
  1519+ bytes            0    0
```

```
-------------------------------------------------------------------------------
Transceiver detail for ethernet-1/3
-------------------------------------------------------------------------------
  Status         : Transceiver is present and operational
  Form factor    : QSFP28
  Channels used  : 4
  Connector type : no-separable-connector
  Vendor         : Mellanox
  Vendor part    : MCP1600-C003
  PMD type       : 100GBASE-CR4 CA-L
  Fault condition: false
  Temperature    : 0
  Voltage        : 0.0000
-------------------------------------------------------------------------------
Transceiver channel detail for ethernet-1/3
-------------------------------------------------------------------------------
  Channel No   Rx Power (dBm)   Tx Power (dBm)   Laser Bias current (mA)
  1            -40.00           -40.00           0.000
  2            -40.00           -40.00           0.000
  3            -40.00           -40.00           0.000
  4            -40.00           -40.00           0.000
===============================================================================
  Subinterface: ethernet-1/3.1
-------------------------------------------------------------------------------
  Oper state      : up
  Down reason     : N/A
  Last change     : 23m14s ago
  Encapsulation   : null
  IP MTU          : 9000
  Last stats clear: never
  IPv4 addr       : 192.57.1.1/31 (static)
  IPv6 addr       : 2001:192:57:1::1/127 (static, preferred)
  IPv6 addr       : fe80::22e0:9cff:fe78:e2e3/64 (link-layer, preferred)
-------------------------------------------------------------------------------
ARP/ND summary for ethernet-1/3.1
-------------------------------------------------------------------------------
  IPv4 ARP entries : 0 static, 0 dynamic
  IPv6 ND  entries : 0 static, 0 dynamic
-------------------------------------------------------------------------------
QOS Policies applied to ethernet-1/3.1
-------------------------------------------------------------------------------
        Summary            In      Out
  IPv4 DSCP classifier   default
  IPv6 DSCP classifier   default
  IPv4 DSCP rewrite                 none
  IPv6 DSCP rewrite                 none
-------------------------------------------------------------------------------
Traffic statistics for ethernet-1/3.1
-------------------------------------------------------------------------------
     Statistics        Rx     Tx
  Packets             52      8
  Octets              14241   828
  Errored packets     0       0
  Discarded packets   2       0
  Forwarded packets   0       0
  Forwarded octets    0       0
  CPM packets         50      8
  CPM octets          14033   828
===============================================================================
```

Example: Display egress queue and VOQ information

To display information about egress queues and Virtual Output Queues (VOQs) for a specific interface and its subinterfaces:

```
# show interface ethernet-1/1 queue-statistics
================================================================================
Interface: ethernet-1/1
--------------------------------------------------------------------------------
  Description     : <None>
  Oper state      : down
  Down reason     : lower-layer-down
  Last change     : 4d14h50m28s ago, No flaps since last clear
  Speed           : 100G
  Flow control    : Rx is disabled, Tx is not supported
  Loopback mode   : false
  MTU             : 9232
  VLAN tagging    : false
  Queues          : 8 output queues supported, 0 used since the last clear
  MAC address     : 68:AB:09:A2:71:B0
  Last stats clear: never
--------------------------------------------------------------------------------
Queue Parameter for for ethernet-1/1
--------------------------------------------------------------------------------
  Queue-id   Scheduling   Weight   PIR %    PIR (kbps)
  0          SP           -        100     98994140.625
  1          SP           -        100     98994140.625
  2          SP           -        100     98994140.625
  3          SP           -        100     98994140.625
  4          SP           -        100     98994140.625
  5          SP           -        100     98994140.625
  6          SP           -        100     98994140.625
  7          SP           -        100     98994140.625
--------------------------------------------------------------------------------
Queue statistics for interface ethernet-1/1, Queue 0 (fc0 traffic)
--------------------------------------------------------------------------------
           Name            Fwd-Octets   Fwd-Pkts   Drop-Octets   Drop-Pkts
  Unicast Egress queue     0            0          0             0
  VOQ 1                    0            0          0             0
  VOQ 2                    0            0          0             0
  VOQ 3                    0            0          0             0
  VOQ 4                    0            0          0             0
  Multicast Egress queue   0            0          0             0
--------------------------------------------------------------------------------
Queue statistics for interface ethernet-1/1, Queue 1 (fc1 traffic)
--------------------------------------------------------------------------------
           Name            Fwd-Octets   Fwd-Pkts   Drop-Octets   Drop-Pkts
  Unicast Egress queue     0            0          0             0
  VOQ 1                    0            0          0             0
  VOQ 2                    0            0          0             0
  VOQ 3                    0            0          0             0
  VOQ 4                    0            0          0             0
  Multicast Egress queue   0            0          0             0
--------------------------------------------------------------------------------
Queue statistics for interface ethernet-1/1, Queue 2 (fc2 traffic)
--------------------------------------------------------------------------------
           Name            Fwd-Octets   Fwd-Pkts   Drop-Octets   Drop-Pkts
  Unicast Egress queue     0            0          0             0
  VOQ 1                    0            0          0             0
  VOQ 2                    0            0          0             0
  VOQ 3                    0            0          0             0
  VOQ 4                    0            0          0             0
  Multicast Egress queue   0            0          0             0
```

```
-------------------------------------------------------------------------------
Queue statistics for interface ethernet-1/1, Queue 3 (fc3 traffic)
-------------------------------------------------------------------------------
           Name            Fwd-Octets   Fwd-Pkts   Drop-Octets   Drop-Pkts
  Unicast Egress queue     0            0          0             0
  VOQ 1                    0            0          0             0
  VOQ 2                    0            0          0             0
  VOQ 3                    0            0          0             0
  VOQ 4                    0            0          0             0
  Multicast Egress queue   0            0          0             0
-------------------------------------------------------------------------------
Queue statistics for interface ethernet-1/1, Queue 4 (fc4 traffic)
-------------------------------------------------------------------------------
           Name            Fwd-Octets   Fwd-Pkts   Drop-Octets   Drop-Pkts
  Unicast Egress queue     0            0          0             0
  VOQ 1                    0            0          0             0
  VOQ 2                    0            0          0             0
  VOQ 3                    0            0          0             0
  VOQ 4                    0            0          0             0
  Multicast Egress queue   0            0          0             0
-------------------------------------------------------------------------------
Queue statistics for interface ethernet-1/1, Queue 5 (fc5 traffic)
-------------------------------------------------------------------------------
           Name            Fwd-Octets   Fwd-Pkts   Drop-Octets   Drop-Pkts
  Unicast Egress queue     0            0          0             0
  VOQ 1                    0            0          0             0
  VOQ 2                    0            0          0             0
  VOQ 3                    0            0          0             0
  VOQ 4                    0            0          0             0
  Multicast Egress queue   0            0          0             0
-------------------------------------------------------------------------------
Queue statistics for interface ethernet-1/1, Queue 6 (fc6 traffic)
-------------------------------------------------------------------------------
           Name            Fwd-Octets   Fwd-Pkts   Drop-Octets   Drop-Pkts
  Unicast Egress queue     0            0          0             0
  VOQ 1                    0            0          0             0
  VOQ 2                    0            0          0             0
  VOQ 3                    0            0          0             0
  VOQ 4                    0            0          0             0
  Multicast Egress queue   0            0          0             0
-------------------------------------------------------------------------------
Queue statistics for interface ethernet-1/1, Queue 7 (fc7 traffic)
-------------------------------------------------------------------------------
           Name            Fwd-Octets   Fwd-Pkts   Drop-Octets   Drop-Pkts
  Unicast Egress queue     0            0          0             0
  VOQ 1                    0            0          0             0
  VOQ 2                    0            0          0             0
  VOQ 3                    0            0          0             0
  VOQ 4                    0            0          0             0
  Multicast Egress queue   0            0          0             0
===============================================================================
```

11 LLDP

The IEEE 802.1ab Link Layer Discovery Protocol (LLDP) standard defines protocol and management elements that are suitable for advertising information to devices attached to the same LAN, for the purpose of populating physical or logical topology and device discovery management information databases.

SR Linux supports the ability to run LLDP on all physical interfaces (both in-band and out-of-band). You can enable or disable LLDP as follows:

- at the global level, using `.system.lldp.admin-state`
- at the interface level, using `.system.lldp.interface{}.admin-state`

If you globally disable LLDP, no interfaces use LLDP. SR Linux does not currently supported TLV suppression.

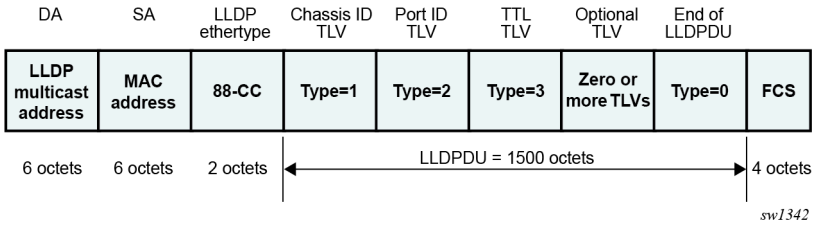
Devices send LLDP information in Ethernet frames from each of their interfaces at a fixed interval. Each frame contains one LLDP Data Unit (LLDPDU), which is a sequence of Type-Length-Value (TLV) structures. LLDP Ethernet frames have the destination MAC address typically set to a special multicast address which 802.1D-compliant bridges do not forward. The EtherType field is set to `0x88cc`.

As shown in Figure 3: LLDP frame structure, each LLDP frame starts with the following mandatory TLVs:

- Chassis-ID
- Port-ID
- Time-to-Live (TTL)

Any number of optional TLVs follow the mandatory TLVs. The frame ends with a special TLV named End of LLDPDU, in which both the `type` and `length` fields are 0.

Figure 3: LLDP frame structure



11.1 LLDP implementation in SR Linux

LLDP is implemented in SR Linux using the `lldp_mgr` application, which controls sending of packets using in-band and out-of-band interfaces (a Linux-native LLDP such as `lldpad` is not used). The `lldp_mgr` crafts packets based on its configuration and forwards them using `xdp`. The SR Linux `xdp` process manages the sending and receiving of frames using in-band and out-of-band ports as follows:

- For ingress in-band, the `xdp` process forwards frames to the active CPM `lldp_mgr` application.
- For egress, the `xdp` process sends frames out the correct interface.

Instead of a hold-time, SR Linux allows you to configure the following two values, which together create the TTL TLV in the LLDPDU (by multiplying the hello-timer by the hold-multiplier):

- hello-timer at `.system.lldp.hello-timer`
- hold-multiplier at `.system.lldp.hold-multiplier`

You cannot configure the following referenced fields:

- The system-name is a leaf reference to `.system.name.host-name`.
- The system-description is auto-generated based on system information. The following is the template for this field.

```
SRLinux-<version> <hostname> <kernel> <build date>
```

Although multiple neighbors are supported under the YANG model, LLDP is not available per VLAN.

11.2 LLDP transmitting system capabilities TLV

Each LLDPDU received and transmitted contains a TLV that references the transmitting system capabilities. The following are the available capabilities:

- SYSTEM_CAPABILITIES_OTHER
- SYSTEM_CAPABILITIES_REPEATER
- SYSTEM_CAPABILITIES_BRIDGE
- SYSTEM_CAPABILITIES_WLAN_AP
- SYSTEM_CAPABILITIES_ROUTER
- SYSTEM_CAPABILITIES_PHONE
- SYSTEM_CAPABILITIES_DOCSIS
- SYSTEM_CAPABILITIES_STATION
- SYSTEM_CAPABILITIES_CVLAN
- SYSTEM_CAPABILITIES_SVLAN
- SYSTEM_CAPABILITIES_TPMR

SR Linux sends only the following capabilities:

- SYSTEM_CAPABILITIES_BRIDGE
- SYSTEM_CAPABILITIES_ROUTER
- SYSTEM_CAPABILITIES_SVLAN

11.3 Configuring LLDP

Procedure

To configure LLDP, enable it at the global level or interface level and configure the hello-timer and hold-multiplier.

Example

The following example shows a basic configuration for LLDP.

```
--{ + running }--[  ]--
# info system lldp
    system {
        lldp {
            admin-state enable
            hello-timer 1
            hold-multiplier 255
            management-address ethernet-1/17.1 {
                type [
                    IPv4
                    IPv6
                ]
            }
            management-address mgmt0.0 {
                type [
                    IPv4
                    IPv6
                ]
            }
            interface ethernet-1/17 {
                admin-state enable
            }
            interface mgmt0 {
                admin-state enable
            }
        }
    }
```

11.4 Displaying LLDP neighbor information

Procedure

LLDP neighbor information is available using a Nokia-provided CLI plugin. You can display LLDP neighbor reports that include information such as the system name, chassis ID, first message, last update, port, and related BGP details. You can also access LLDP neighbor reports for a particular interface.

Example

```
--{ running }--[  ]--
# show system lldp neighbor
  +----------+----------+----------+----------+----------+----------+----------+
  |   Name   | Neighbor | Neighbor | Neighbor | Neighbor | Neighbor | Neighbor |
  |          |          |  System  | Chassis  |  First   |   Last   |   Port   |
  |          |          |   Name   |    ID    | Message  |  Update  |          |
  +==========+==========+==========+==========+==========+==========+==========+
  | ethernet | 00:01:01 | 3-node-s | 00:01:01 | 7 days   | 6        | ethernet |
  | -1/1     | :FF:00:0 | rlinux-A | :FF:00:0 | ago      | minutes  | -1/1     |
  |          | 0        |          | 0        |          | ago      |          |
  | ethernet | 00:01:03 | 3-node-s | 00:01:03 | 7 days   | 6        | ethernet |
  | -1/3     | :FF:00:0 | rlinux-C | :FF:00:0 | ago      | minutes  | -1/3     |
  |          | 0        |          | 0        |          | ago      |          |
  +----------+----------+----------+----------+----------+----------+----------+
```

See the *SR Linux System Management Guide*, section Pre-defined show reports and System show reports for more information.

11.5 Displaying LLDP statistics

Procedure

You can display aggregate statistics and per-interface statistics for LLDP including frame discards, frame-in and frame-out numbers, last clear, and TLV accepted numbers.

To display statistics use the **info from state** command in candidate or running mode or the **info** command in state mode.

Example: Display LLDP statistics

```
--{ + running }--[  ]--
# info from state system lldp statistics
    system {
        lldp {
            statistics {
                frame-in 516
                frame-out 2565
                last-clear "3 minutes ago"
                tlv-accepted 5680
            }
        }
    }
```

Example: Display LLDP detail statistics

```
--{ + running }--[  ]--
# info detail from state system lldp statistics
    system {
        lldp {
            statistics {
                frame-in 22
                frame-out 88
                last-clear 2022-10-24T18:57:26.468Z
                tlv-accepted 274
            }
        }
    }
```

Example: Display LLDP detail statistics for the management interface

```
--{ + running }--[  ]--
# info detail from state system lldp interface mgmt0 statistics
    system {
        lldp {
            interface mgmt0 {
                statistics {
                    frame-in 0
                    frame-out 344426
                    frame-error-in 0
                    frame-discard 120
                    tlv-discard 0
                    tlv-unknown 0
                    frame-error-out 1
```

```
                }
            }
        }
    }
```

Example: Display LLDP statistics for an Ethernet interface

```
--{ + running }--[  ]--
# info from state system lldp interface ethernet-1/17 statistics
    system {
        lldp {
            interface ethernet-1/17 {
                statistics {
                    frame-in 331
                    frame-out 333
                    last-clear "5 minutes ago"
                }
            }
        }
    }
```

Example: Display LLDP detail statistics for an Ethernet interface

```
--{ + running }--[  ]--
# info detail from state system lldp interface ethernet-1/17 statistics
    system {
        lldp {
            interface ethernet-1/17 {
                statistics {
                    frame-in 5
                    frame-out 5
                    last-clear 2022-10-24T18:58:44.534Z
                }
            }
        }
    }
```

11.6 Clearing LLDP statistics

Procedure

You can clear aggregate statistics and per-interface statistics for LLDP. Clearing the statistics counters resets all statistics to 0 and populates the `last-clear` field with the current timestamp.

Use the following commands in running, candidate, or state mode to clear LLDP statistics counters.

```
tools system lldp statistics clear
tools system lldp interface name statistics clear
```

Example: Clear LLDP statistics

```
--{ + running }--[  ]--
# tools system lldp statistics clear
--{ + running }--[  ]--
# info from state system lldp statistics
    system {
        lldp {
```

```
            statistics {
                frame-in 2
                frame-out 11
                last-clear now
                tlv-accepted 28
            }
        }
    }
```

Example: Clear management-interface statistics

```
--{ + running }--[  ]--
# tools system lldp interface mgmt0 statistics clear
--{ + running }--[  ]--
# info detail from state system lldp interface mgmt0 statistics
    system {
        lldp {
            interface mgmt0 {
                statistics {
                    frame-out 1
                    last-clear 2022-10-24T18:59:46.291Z
                }
            }
        }
    }
```

Example: Clear Ethernet interface statistics

```
--{ + running }--[  ]--
# tools system lldp interface ethernet-1/17 statistics clear
--{ + running }--[  ]--
# info from state system lldp interface ethernet-1/17 statistics
    system {
        lldp {
            interface ethernet-1/17 {
                statistics {
                    frame-in 1
                    frame-out 1
                    last-clear now
                }
            }
        }
    }
```

12 LAG

A Link Aggregation Group (LAG), based on the IEEE 802.1ax standard (formerly 802.3ad), increases the bandwidth available between two network devices, depending on the number of links installed. A LAG also provides redundancy if one or more links participating in the LAG fail. All physical links in a LAG combine to form one logical interface.

Packet sequencing is maintained for individual sessions. The hashing algorithm deployed by SR Linux is based on the type of traffic transported to ensure that all traffic in a flow remains in sequence, while providing effective load sharing across the links in the LAG.

LAGs can be either statically configured, or formed dynamically with Link Aggregation Control Protocol (LACP). Load sharing is executed in hardware, which provides line rate forwarding for all port types. A LAG can consist of ports of the same speed, as well as ports of mixed speed; however, the active links would be only those whose port speed matches the configured **member-speed** parameter for the LAG instance.

12.1 Min-link threshold

SR Linux supports configuring a min-link threshold for a LAG, which sets the minimum number of member links that must be active in order for the LAG to be operationally up. If the number of active links falls below this threshold, the entire LAG is brought operationally down.

If the min-link threshold is crossed, the active member links are maintained, including continuing to run LACP on links where it is configured, but the LAG is held out of forwarding state. When the number of active links reaches or exceeds the min-link threshold, the LAG is brought back up operationally.

12.2 LACP

LACP, defined by the IEEE 802.3ad standard, specifies a method for two devices to establish and maintain LAGs. When LACP is enabled, SR Linux can automatically associate LACP-compatible ports into a LAG. All non-failing links in a LAG are active, and traffic is load-balanced across the active links.

When LACP is enabled, LACP changes are visible through traps and log messages logged against the LAG.

12.2.1 LACP fallback

LACP fallback allows one or more designated links of an LACP controlled LAG to go into forwarding mode if LACP is not yet operational after a configured timeout period.

SR Linux supports LACP fallback in static mode. In static mode, a single designated LAG member goes into forwarding mode if LACP is not operational after the timeout period.

LACP fallback is configured by selecting the mode and fallback timeout (seconds). If the LAG receives no PDUs and the timeout period expires, the configured fallback mode is enabled. If any member link in the LAG receives a PDU, the fallback mode is immediately disabled.

12.3 LAG configuration

To configure a LAG, you specify LAG parameters within the context of a LAG interface, then associate Ethernet interfaces with the LAG interface.

The MAC address of the LAG should be a unique value taken from the chassis MAC address pool.

Member links in the LAG can be associated statically or dynamically.

- Static links are explicitly associated with the LAG within the configuration of the LAG instance.
- Dynamic links are associated with the LAG using LACP.

A LAG instance can consist of static links only or dynamic links only.

If an Ethernet interface is associated with a LAG interface, the following parameters must be the same for all associated Ethernet ports:

- **flow-control**
- **port-speed**
- **aggregate-id**

The following example shows the configuration for a LAG consisting of three member links.

```
--{ * candidate shared default }--[  ]--
# info interface *
    interface ethernet-1/1 {
        admin-state enable
        ethernet {
            aggregate-id lag1
        }
    }
    interface ethernet-1/2 {
        admin-state enable
        ethernet {
            aggregate-id lag1
        }
    }
    interface ethernet-1/3 {
        admin-state enable
        ethernet {
            aggregate-id lag1
        }
    }
    interface lag1 {
        subinterface 1 {
            admin-state enable
        }
        lag {
            lag-type static
            min-links 2
        }
    }
```

12.3.1 Configuring the min-link threshold

Procedure

You can configure the min-link threshold for a LAG, which specifies the minimum number of member links that must be active in order for the LAG to be operationally up. If the number of active links falls below this threshold, the entire LAG is brought operationally down.

Example

The following example configures the min-link threshold for a LAG to be 4. If the number of active links in the LAG drops below 4, the LAG is taken operationally down.

```
--{ * candidate shared default }--[  ]--
# info interface lag1
    interface lag1 {
        lag {
            min-links 4
        }
    }
```

After the LAG has been taken operationally down because of crossing the min-link threshold, if the number active links in the LAG subsequently reaches 4 or higher, the LAG is brought operationally up. The default for the min-link threshold is 0 (disabled).

12.3.2 Configuring LACP and LACP fallback

Procedure

When you enable LACP, SR Linux can automatically associate LACP-compatible ports into a LAG. LACP should be configured in `ACTIVE` mode only if LACP Fallback is also configured.

Example: Configure LACP to run on an interface

The following example configures LACP to run on an interface, which can dynamically become a member of a LAG:

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1 lag
    interface ethernet-1/1 {
        lag {
            lag-type lacp
            min-links 1
            member-speed 100G
            lacp-fallback-mode static
            lacp-fallback-timeout 4
            lacp {
                interval FAST
                lacp-mode ACTIVE
            }
        }
    }
```

In this example, the LACP interval is set to **FAST**, which causes LACP messages to be sent every second. The **SLOW** option for LACP interval causes LACP messages to be sent every 30 seconds.

Example: Enable LACP fallback mode for a LAG

The following example enables LACP fallback mode for a LAG, which allows a single designated LAG member to go into forwarding mode if LACP is not operational after the timeout period.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1 lag
    interface ethernet-1/1 {
        lag {
            lacp-fallback-mode static
            lacp-fallback-timeout 60
        }
    }
```

The LACP fallback timeout range is 4 to 3600 seconds when the LACP interval is **FAST**, and 90 to 3600 seconds when LACP interval is **SLOW**.

Example: Enable LACP port priority

The following example enables LACP port priority. When LACP fallback is triggered in static mode, one of the member-links goes into a forwarding state that can be influenced using LACP port priority.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1 ethernet
    interface ethernet-1/1 {
        ethernet {
            aggregate-id lag1
            lacp-port-priority 1
            port-speed 25G
            hw-mac-address 00:01:02:FF:00:01
        }
    }
```

12.3.3 Configuring forwarding viability for LAG member links

Procedure

By default, all interfaces configured in a LAG are capable of forwarding traffic to the other end of the LAG, assuming all other LAG and port attributes allow it (port and LACP state). You can optionally configure individual LAG members to be non-viable for forwarding traffic to the other end of the LAG link.

When a LAG member is configured as non-viable for forwarding traffic, the interface is not used for the transmission of traffic over the LAG, but is still able to process traffic it receives on the associated member link. In addition, Layer 2 protocols such as LLDP, LACP, and micro-BFD continue to be sent and processed over the non-forwarding-viable LAG member.

Example

The following example configures a LAG member to be non-viable for forwarding traffic across a LAG link. The interface can still receive traffic on the LAG link and participate in Layer 2 functions, but does not transmit packets to the other end of the LAG.

```
--{ candidate shared default }--[  ]--
# info interface ethernet-1/1 ethernet
    interface ethernet-1/1 {
        ethernet {
            aggregate-id lag1
            forwarding-viable false
```

```
        }
    }
```

Note: If the **forwarding-viable** command is explicitly configured for a LAG member, you must delete the **forwarding-viable** command to remove the LAG member from the LAG instance.

12.4 Displaying LAG interface statistics

Procedure

To display statistics for a LAG interface, use the **info from state** command in candidate or running mode, or the **info** command in state mode.

Example

```
--{ candidate shared default }--[  ]--
# info from state interface lag1 statistics
    interface lag1 {
        statistics {
            in-octets 0
            in-unicast-packets 0
            in-broadcast-packets 0
            in-multicast-packets 0
            in-error-packets 0
            in-fcs-error-packets 0
            out-octets 7168
            out-unicast-packets 0
            out-broadcast-packets 0
            out-multicast-packets 56
            out-error-packets 0
            last-clear 2020-06-09T21:58:40.919Z
        }
    }
```

12.4.1 Clearing LAG interface statistics

Procedure

You can clear the statistics counters for a specified LAG interface.

Example: Clear statistics counters for a LAG interface

```
--{ candidate shared default }--[  ]--
# tools interface lag1 statistics clear
/interface[name=lag1]:
    interface lag1 statistics cleared
```

Example: Clear statistics for a LAG interface and all member links

```
--{ candidate shared default }--[  ]--
# tools interface lag1 statistics clear include-members
/interface[name=lag1]:
    interface lag1 and all member interfaces statistics cleared
```

13 Breakout ports

Breakout functionality is supported in several platforms in the 7220 product family. Breakout functionality is supported in several platforms in the 7220 product family. One such example is the QSFP28 connector ports (1-31) in the 7220 IXR-D3L and ports (9, 15, 16, and 30-36) in the 7220 IXR-D4 can be broken out to either 4x10G or 4x25G in breakout mode.

In this example, breakout ports are named using the following format:

`ethernet-`*`slot/port/breakout-port`*

For example, if interface `ethernet 1/3` is enabled for breakout mode, its breakout ports are named as follows:

- `ethernet 1/3/1`
- `ethernet 1/3/2`
- `ethernet 1/3/3`
- `ethernet 1/3/4`

When an interface is operating in breakout mode, it is considered a breakout connector, and not an Ethernet port. Some features that are configurable on an Ethernet port do not apply to a breakout connector. The following parameters cannot be configured on an interface operating as a breakout connector:

- `mtu`
- `loopback-mode`
- `aggregate-id`
- `auto-negotiate`
- `duplex-mode`
- `flow-control receive`
- `flow-control transmit`
- `lacp-port-priority`
- `port-speed`
- `standby-signaling`
- `reload-delay`
- `hold-time`
- `storm-control`
- `vlan-tagging`
- `subinterface`
- `lag`
- `qos`

- `sflow`
- `transceiver`

When the **admin-state** parameter for a breakout connector is set to **disable**, it causes its breakout ports to be shut down. In this case, the output from the **info from state** command lists the oper-down-reason for the breakout ports as connector-down.

The **port-speed** setting is not configurable for a breakout port. The speed of the breakout port is determined by the **channel-speed** setting for the breakout connector.

Note the following when configuring the **transceiver** parameter for a breakout port:

- The **tx-laser** setting affects only the individual breakout port. If the installed transceiver supports per-channel disabling of the TX laser then configuring `tx-laser = false` causes the state of the breakout port to be oper-down.

  If the installed transceiver does not support per-channel disabling of the TX laser, then the state of the breakout port remains oper-up and **info from state** displays `tx-laser=true`.
- If `ddm-events = true` is configured for any breakout port, then the system generates warning logs for temperature and voltage of the overall transceiver/connector.

  If `ddm-events = false` for any breakout port, the system suppresses warning logs for input-power, output-power, and laser-bias-current for that specific port/laser.
- The configured `forward-error-correction` algorithm applies only to the individual breakout port.

13.1 Configuring breakout mode for an interface

Procedure

To enable breakout ports, you enable breakout mode for an interface and configure breakout ports for the interface.

Example

The following is an example of configuring an interface for breakout-mode and enabling breakout ports on the interface.

```
--{ candidate shared default }--[  ]--
# info interface ethernet-1/3*
    interface ethernet-1/3 {
        admin-state enable
        description "Breakout connector"
        breakout-mode {
            num-breakout-ports 4
            breakout-port-speed 25G
            }
        }
    }
    interface ethernet-1/3/1 {
        admin-state enable
        description "Breakout port 1"
        subinterface 1 {
            admin-state enable
            ipv4 {
                admin-state enable
                address 192.168.12.1/30 {
                }
```

```
            }
        }
    }
    interface ethernet-1/3/2 {
        admin-state enable
        description "Breakout port 2"
        subinterface 1 {
            admin-state enable
            ipv4 {
                admin-state enable
                address 192.168.12.5/30 {
                }
            }
        }
    }
    interface ethernet-1/3/3 {
        admin-state enable
        description "Breakout port 3"
        subinterface 1 {
            admin-state enable
            ipv4 {
                admin-state enable
                address 192.168.12.9/30 {
                }
            }
        }
    }
    interface ethernet-1/3/4 {
        admin-state enable
        description "Breakout port 4"
        subinterface 1 {
            admin-state enable
            ipv4 {
                admin-state enable
                address 192.168.12.13/30 {
                }
            }
        }
    }
```

14 PHY-to-port-group mapping (7220 IXR-D2 and 7220 IXR-D2L only)

On 7220 IXR-D2 and 7220 IXR-D2L devices there are 12 port groups, consisting of 4 ports each. The group a port belongs to corresponds to the PHY connected to the port; the ports in a port group are all connected to the same PHY.

On 7220 IXR-D2 devices, the Ethernet ports are grouped using the following mapping:

Figure 4: Port-group mapping on 7220 IXR-D2

1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47	49	51	53	55
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54	56

In this mapping, ports 1, 2, 3, and 4 belong to one port group, ports 5, 6, 7, and 8 belong to another port group, and so on.

On 7220 IXR-D2L devices, the Ethernet ports are grouped using the following mapping:

Figure 5: Port-group mapping on 7220 IXR-D2L

1	4	7	10	13	16	19	22	25	28	31	34	37	40	43	46			
2	5	8	11	14	17	20	23	26	29	32	35	38	41	44	47	49	51	53	55
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	50	52	54	56

In this mapping, ports 1, 2, 3, and 6 belong to one port group, ports 4, 5, 7, and 9 belong to another port group, and so on.

14.1 Displaying the members of a port group

Procedure

To list the members of the port group to which a specified port belongs, use the **info from state interface phy-group-members** command. All of the ports in the group are connected to the same PHY on the 7220 IXR-D2/D2L device.

Example

The following example displays the members of the port group to which interface `ethernet 1/1` on a 7220 IXR-D2L device belongs.

```
--{ running }--[  ]--
# info from state interface ethernet-1/1 phy-group-members
    interface ethernet-1/1 {
        phy-group-members [
            ethernet-1/1
            ethernet-1/2
            ethernet-1/3
            ethernet-1/6
```

```
        ]
    }
```

15 IPv4 ARP and IPv6 ND

Address Resolution Protocol (ARP) allows an IPv4 host or router to learn the link-layer (MAC) address that is associated with a neighbor's IPv4 address. IPv4 ARP also provides address conflict detection.

Similarly, the Neighbor Discovery (ND) protocol allows an IPv6 host or router to learn the link-layer address that is associated with a neighbor's IPv6 address. IPv6 ND also supports duplicate address detection (DAD) and neighbor unreachability detection (NUD).

In SR Linux the arp_nd_mgr is the software process that handles the sending and receiving of ARP and ND messages.

Interaction between SR Linux and the underlying Linux OS

The underlying Linux OS uses its own ARP/ND stack to send ARP requests and neighbor solicitations as required by native Linux applications. The SR Linux arp_nd_mgr intercepts each request and, if it has a matching neighbor entry, replies to it immediately. Otherwise arp_nd_mgr sends the request and processes the neighbor responses, which it then forwards back to the underlying Linux OS.

In addition, if **network-instance protocols linux export-neighbors** is set to **true**, linux_mgr adds all neighbors that are known to arp_nd_mgr as static neighbors in the underlying Linux OS.

15.1 IPv4 ARP

ARP allows a router to determine the next-hop MAC address for a particular destination IPv4 address. SR Linux provides the following ARP support on IPv4 routed subinterfaces:

- static ARP entries
- dynamic ARP entries with configurable timeout per subinterface
- EVPN entries (on platforms where EVPN is enabled)
- address conflict detection (ACD) per RFC 5227

Static ARP entries

A static ARP entry associates an IPv4 address with a MAC address on a subinterface. When the configuration is committed, arp_nd_mgr interacts with the Nokia eXtensible Data Path (XDP) hardware abstraction layer to add the ARP entry into the relevant hardware tables.

Dynamic ARP entries

SR Linux creates dynamic ARP requests in the following cases:

- **Routing next-hop groups**: When you create a static next-hop-group with an IPv4 next-hop (or when fib_mgr creates a dynamic next-hop-group with an IPv4 next-hop) and no ARP entry exists for the next-hop, SR Linux immediately sends an ARP request for the next-hop, even without a traffic trigger.
- **Locally connected hosts**: When the system receives an IPv4 packet and the destination is a host address on a local subnet for which there is no ARP entry, arp_nd_mgr sends an ARP request for the

destination. While the system waits for a reply, additional packets destined for the destination may be buffered.

- **Linux applications**: When a Linux application sends an ARP request for an address, arp_nd_mgr intercepts the message and if no matching ARP entry exists, arp_nd_mgr creates its own ARP request for the address.

After arp_nd_mgr sends an ARP request, it performs the following steps:

1. If no ARP reply is received, the ARP entry is put on the retry list that the system revisits every 30 seconds. The ARP entry does not appear in **info from state** output yet.
2. When the system receives an ARP reply, a dynamic entry is programmed and its expiration timeout is based on the subinterface configuration; the default is 4 hours. The expiration timeout is reset whenever the subinterface receives any ARP packet from the associated source address.
3. 30 seconds before the expiration timeout ends, arp_nd_mgr considers the ARP entry to be stale and automatically sends a new ARP request for the address. If it receives a response, the expiration timeout is reset. If no response is received when the expiration timeout ends, the ARP entry is deleted and removed from **info from state**.

   For expired next-hop route addresses, the system periodically resends new ARP requests to attempt to resolve those entries.

Address conflict detection

By default, address conflict detection is enabled on every router subinterface, but it can be disabled per subinterface.

15.1.1 Configuring static ARP entries

Procedure

To create a static ARP entry on a subinterface, use the **ipv4 arp neighbor** *ipv4-address* **link-layer-address** *mac-address* command to associate the IPv4 address of a neighbor with its MAC address.

Example: Configure a static ARP entry

```
--{ + candidate shared default }--[ ]--
# info interface ethernet-1/1 subinterface 2 ipv4 arp
    interface ethernet-1/1 {
        subinterface 2 {
            ipv4 {
                arp {
                    timeout 28800
                    neighbor 192.168.10.3 {
                        link-layer-address 00:00:5E:00:53:EF
                    }
                }
            }
        }
    }
```

15.1.2 Configuring dynamic ARP timeout

Procedure

For dynamic ARP entries, you can optionally modify the expiration timeout value on a subinterface using the **ipv4 arp timeout** command. The time remaining for existing dynamic ARP entries on the subinterface is not affected by this update.

Example: Configure dynamic ARP timeout on a subinterface

```
--{ + candidate shared default }--[ ]--
# info interface ethernet-1/1 subinterface 2 ipv4 arp timeout
    interface ethernet-1/1 {
        subinterface 2 {
            ipv4 {
                arp {
                    timeout 28800
                }
            }
        }
    }
```

15.1.3 Disabling ARP address conflict detection

Procedure

By default, address conflict detection (ACD) is enabled on every router subinterface, but you can disable it for a subinterface by setting **ipv4 arp duplicate-address-detection** to **false**.

However, if the subinterface is configured as a DHCPv4 client, ACD is always performed, regardless of this per-subinterface setting.

Example: Disable ARP address conflict detection on a subinterface

```
--{ + candidate shared default }--[  ]--
# info interface ethernet-1/1 subinterface 1 ipv4 arp
    interface ethernet-1/1 {
        subinterface 1 {
            ipv4 {
                arp {
                    duplicate-address-detection false
                }
            }
        }
    }
```

15.2 IPv6 Neighbor Discovery

Neighbor Discovery (ND) protocol allows a router to determine the next-hop MAC address for a particular destination IPv6 address. SR Linux provides the following ND support on IPv6 routed subinterfaces:

- support for all 5 neighbor states: incomplete, reachable, stale, delay, and probe
- static neighbor cache entries
- dynamic neighbor cache entries with configurable timeout per subinterface

- duplicate address detection
- neighbor unreachability detection

Static IPv6 Neighbor Entries

A static neighbor entry associates an IPv6 address with a MAC address on a subinterface. When the configuration is committed, arp_nd_mgr interacts with XDP to add the neighbor entry into the relevant hardware tables.

IPv6 Neighbor Entries for Routing Next-Hops

When you create a static next-hop-group with an IPv6 next-hop (or when fib_mgr creates a dynamic next-hop-group with an IPv6 next-hop) and no neighbor entry exists for that address, arp_nd_mgr immediately sends a neighbor solicitation request for the address, even without a traffic trigger. The neighbor solicitation process is as follows:

- If no neighbor advertisement is received for the target address, the neighbor solicitation message is retransmitted every 1 second (with some randomization). The retransmit interval is not configurable. The neighbor entry does not appear in **info from state** yet.
- If a neighbor advertisement is received for the target address, a dynamic entry is programmed with an initial state of `reachable`. The state changes to `stale` after the reachable time expires. The reachable time is configurable per subinterface with a default of 30 seconds.
- While the neighbor is `stale`, the system makes no attempt to confirm reachability using neighbor solicitations, even if there is traffic destined for the target address. The neighbor state changes from `stale` to `delay` (and subsequently `probe`) after the stale time expires. Stale time is configurable per subinterface with a default of 14 400 seconds. The arp_nd_mgr attempts to refresh the neighbor entry by sending a neighbor solicitation (and retransmitting twice if required). If no response is received, the neighbor entry is deleted and removed from **info from state**.

IPv6 neighbor limit on subinterfaces

You can set a limit on the number of IPv6 neighbors that a subinterface can learn. The following considerations apply:

- The limit only applies to dynamic neighbors. Static and EVPN neighbors do not count towards the limit and can still be added when the dynamic neighbor limit is exceeded.
- If a subinterface already has a number of dynamic neighbor entries and you set the neighbor limit to below the current number of entries, the router does not remove the exceeding entries. Existing neighbors are still refreshed. The limit only applies to new learned neighbors.
- The system provides two log events that warn about the number of entries exceeding the configured threshold.

15.2.1 Configuring static ND entries

Procedure

To create a static ND entry on a subinterface, use the **ipv6 neighbor discovery neighbor** *ipv6-address* **link-layer-address** *mac-address* command to associate the IPv6 address of a neighbor with its MAC address.

Example: Configure a static ND entry for a subinterface

```
--{ +* candidate shared default }--[  ]--
# info interface ethernet-1/1 subinterface 2 ipv6 neighbor-discovery neighbor 2001:db8::1
    interface ethernet-1/1 {
        subinterface 2 {
            ipv6 {
                neighbor-discovery {
                    neighbor 2001:db8::1 {
                        link-layer-address 00:00:5E:00:53:AF
                    }
                }
            }
        }
    }
```

15.2.2 Configuring ND reachable time and stale time

Procedure

If the system receives a neighbor advertisement with a target address, a dynamic entry is programmed with an initial state of `reachable`. The state changes to `stale` after **reachable-time** seconds. The **reachable-time** is configurable per subinterface with a default of 30 seconds. While the neighbor is `stale`, the system makes no attempt to confirm reachability using neighbor solicitations, even if there is traffic destined for the target address. The neighbor state changes from `stale` to `delay` (and subsequently `probe`) after the **stale-time** expires. The **stale-time** is configurable per subinterface with a default of 14 400 seconds.

Example: Configure ND reachable time and stale time on a subinterface

```
--{ + candidate shared default }--[  ]--
# info interface ethernet-1/1 subinterface 1 ipv6 neighbor-discovery
    interface ethernet-1/1 {
        subinterface 1 {
            ipv6 {
                neighbor-discovery {
                    reachable-time 50
                    stale-time 28800
                }
            }
        }
    }
```

15.2.3 Configuring IPv6 neighbor limit on subinterfaces

Procedure

To set a limit for IPv6 neighbors, use the **subinterface ipv6 neighbor-discovery limit** command, which has the following configurable parameters:

- **max-entries**: Sets the maximum number of neighbor entries allowed on the subinterface.
- **log-only**: Defines the action taken when the subinterface exceeds the **max-entries** limit. When set to **true**, the system keeps learning entries on the subinterface and only logs an event.

- **warning-threshold-pct**: Sets the percentage of max-entries that triggers a log event indicating the limit is approaching. The default value is 90 percent of the configured **max-entries** limit. The event is logged only the first time a neighbor exceeds the limit, and the condition is cleared if the limit falls back below the threshold.

  The threshold value is calculated as follows:

  (**max-entries** * **warning-threshold-pct**)/100

  However, the system rounds the result down. Therefore, a **max-entries** setting of **2** provides the same result whether the **warning-threshold-pct** is **90** percent or **50** percent:

  2*90/100 = 2*50/100 = 1

Example: Configure IPv6 neighbor limit on a subinterface

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1 subinterface 3 ipv6 neighbor-discovery
    interface ethernet-1/1 {
        subinterface 3 {
            ipv6 {
                neighbor-discovery {
                    limit {
                        max-entries 10
                        log-only false
                        warning-threshold-pct 85
                    }
                }
            }
        }
    }
```

16 DHCP relay

DHCP relay refers to the router's ability to act as an intermediary between DHCP clients requesting configuration parameters, such as a network address, and DHCP servers when the DHCP clients and DHCP servers are not attached to the same broadcast domain, or do not share the same IPv6 link (in the case of DHCPv6).

SR Linux supports DHCP relay for IRB subinterfaces and Layer 3 subinterfaces. Up to 8 DHCP or DHCPv6 servers are supported. The DHCP relay maximum packet size (including option 82 and vendor-specific options) is capped at 1500 bytes to avoid fragmentation on the Ethernet segment end attached to the DHCP server.

When DHCP relay is enabled for a subinterface, and a DHCP client initiates a request for configuration parameters, the router accepts the DHCP client's request and relays it to the remote DHCP server, which sends back the configuration parameters. The router relays the configuration parameters to the client.

The DHCP server network can be in the same IP-VRF network-instance of the Layer 3 subinterfaces that require DHCP relay (see Figure 6: DHCP relay for IRB and Layer 3 subinterfaces), or it can be in a different IP-VRF network-instance or the default network instance (see Figure 7: DHCP relay using different IP-VRF or default network-instance).

SR Linux supports DHCP relay for IPv4 and IPv6. This guide refers to DHCP for IPv4 as DHCP, and DHCP for IPv6 as DHCPv6.

Figure 6: DHCP relay for IRB and Layer 3 subinterfaces

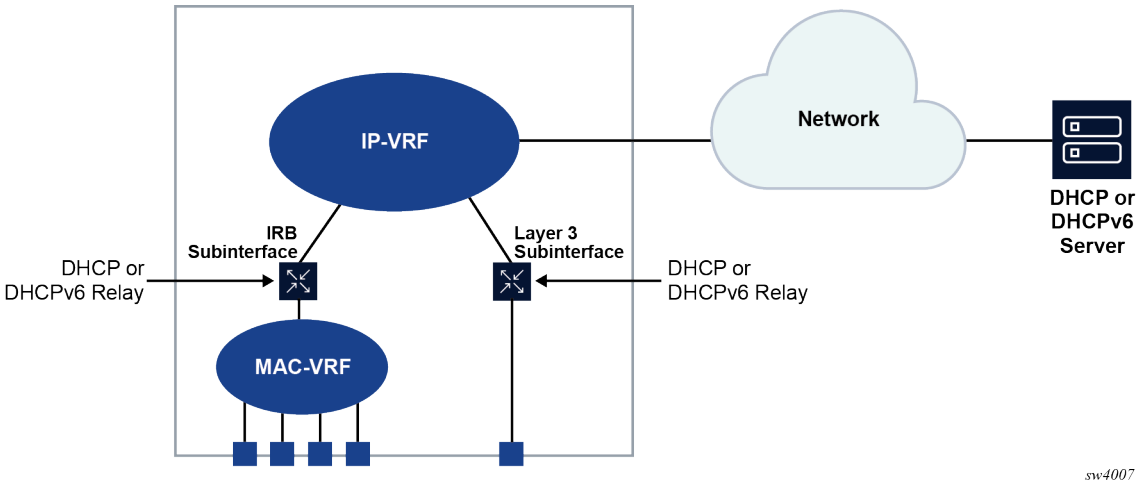
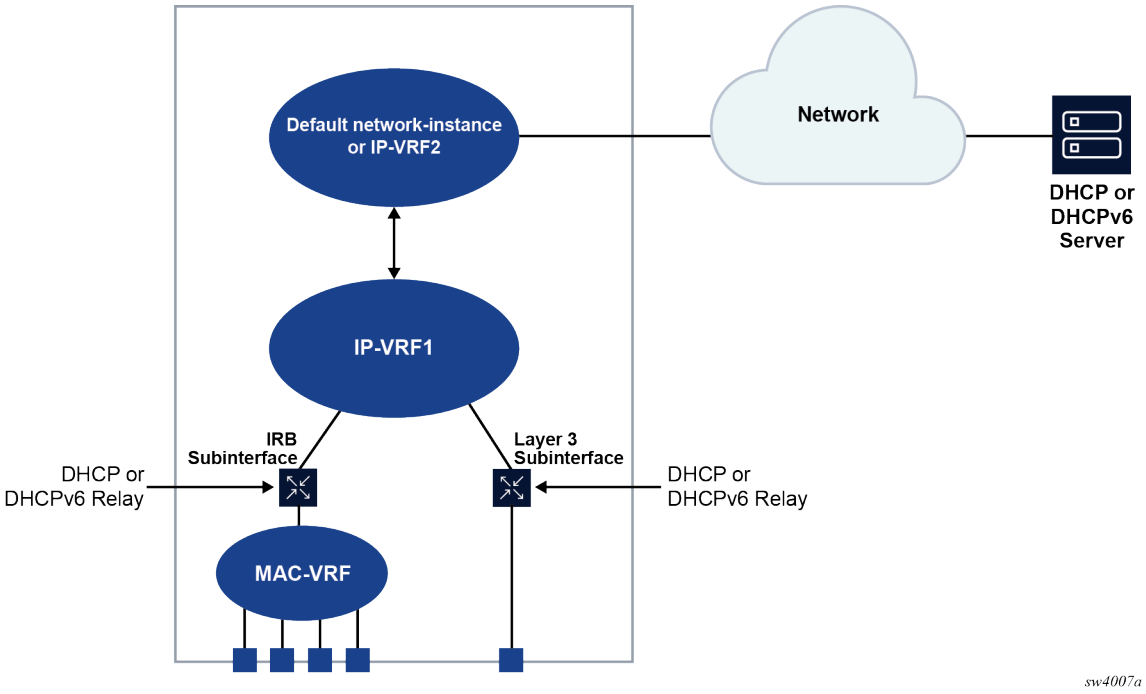


Figure 7: DHCP relay using different IP-VRF or default network-instance



16.1 DHCP relay for IPv4

When DHCP relay is enabled, the router intercepts DHCP broadcast packets and unicasts them to a specified DHCP server for handling. By default, the source address for DHCP packets relayed to the server (GIADDR) is the IP address of the ingress subinterface where the DHCP relay agent is enabled, although a different GIADDR can be specified if necessary.

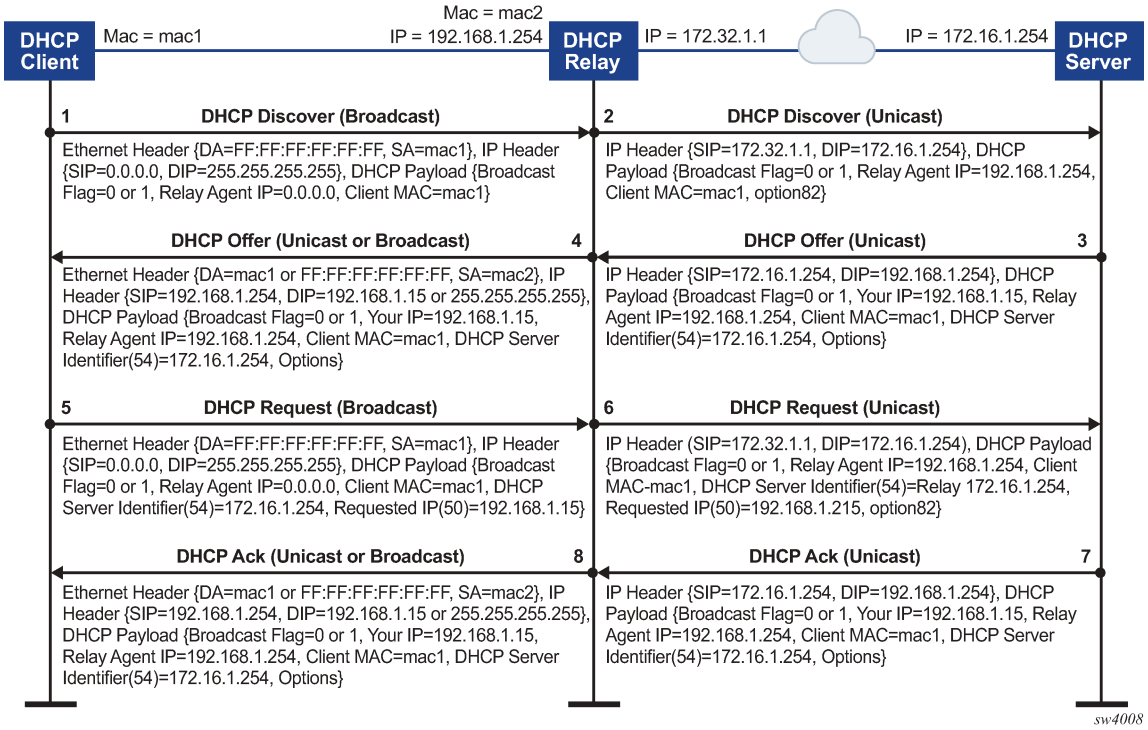
SR Linux supports DHCP option 82, the Relay Information Option, specified in RFC 3046, which allows the router to append information to DHCP requests relayed to the DHCP server, identifying where the original DHCP request came from. DHCP option 82 includes two sub-options: circuit-id and remote-id.

When configured to do so, SR Linux includes the following information in the circuit-id and remote-id sub-options of DHCP option 82:

- For circuit-id, the system_name/VRF_instance/sub-interface_id:vlan_id of the ingress subinterface where the relay agent is enabled that receives the DHCP Discover message from the DHCP client.
- For remote-id, the MAC address of the DHCP client.

Figure 8: DHCP message flow for IPv4 address allocation shows an example of the discovery, offer, request, and acknowledgment (DORA) message flow that occurs when DHCP relay assigns an address to a DHCP client.

Figure 8: DHCP message flow for IPv4 address allocation



The DORA message flow shown in Figure 8: DHCP message flow for IPv4 address allocation works as follows:

1. The DHCP client sends a DHCP Discover (broadcast) message with the following values:
   - DA = FF:FF:FF:FF:FF:FF (broadcast)
   - SA= client MAC
   - SIP = 0.0.0.0
   - DIP = 255.255.255.255
   - Source UDP port = 68
   - Destination UDP port = 67

   The DHCP payload has the following values:
   - Broadcast flag = 1 (broadcast) or 0 (unicast)
   - Relay agent IP = 0.0.0.0
   - Client MAC = mac1
   - Parameter request list (option 55) which lists the required items from the DHCP server to be sent along with the IP address like subnet mask, router (gateway), and others
2. The DHCP relay agent relays the DHCP Discover message toward the DHCP server (unicast). If configured to do so, information is added for the circuit ID and remote ID sub-options in DHCP option 82. The relayed packet is unicast toward the DHCP servers with the following values:
   - SIP = outgoing interface IP address by default. If the source-address is configured, the relayed packet instead has SIP = configured source-address

- UDP source port = 67
- UDP destination port = 67

The DHCP payload has the following values:

- Broadcast = 1 (broadcast) or 0 (unicast)
- Relay agent IP (giaddr) = IP address of the ingress sub-interface where the relay agent is enabled
- Client MAC = mac1
- Relay agent information (option 82)

3. The DHCP server assigns an IP address to the DHCP client, based on information in the GIADDR or in option 82, if configured to do so. The DHCP server sends a DHCP Offer message to the DHCP relay agent (unicast). The DHCP Offer message includes the IP address assigned to the DHCP client based on information in the GIADDR or in option 82.

   The DHCP Offer packet is unicast with the following values:

   - SIP = DHCP IP address
   - DIP = giaddr
   - UDP source port = 67
   - UDP destination port = 67

   The DHCP payload has the following values:

   - Broadcast flag = 1 (broadcast) or 0 (unicast).
   - Your (client) IP = IP address assigned by DHCP server
   - Agent IP = giaddr
   - Client MAC = mac1
   - DHCP identifier = DHCP server IP address
   - Option 82 (echoed back, and based on DHCP server configuration)
   - IP address Lease time (option 51)
   - Subnet mask (option 1)
   - Router (gateway) (option 3)
   - Others (DNS, Renewal Time value, Rebinding Time value, and so on)

4. The DHCP relay agent relays the DHCP Offer message to the DHCP client (either broadcast or unicast, based on the broadcast flag sent by the client).

   The DHCP Offer message is relayed from the DHCP server toward the client with the following values:

   - DA = FF:FF:FF:FF:FF:FF (broadcast) OR Client MAC(unicast)
   - SIP = sub-interface IP address toward the client where DHCP relay agent is enabled
   - DIP = 255.255.255.255 (broadcast) OR Your (client) IP address (unicast)
   - Source UDP port = 67
   - Destination UDP port = 68

   The relay agent relays the DHCP Offer toward the client without option 82. It strips off option 82 if echoed back from DHCP server.

   The DHCP payload has the following values:

- Broadcast flag = 1 (broadcast) or 0 (unicast).
- Your (client) IP = IP address assigned by DHCP server
- Agent IP = giaddr
- Client MAC = mac1
- DHCP identifier = DHCP server IP address
- Option 82 (echoed back, and based on DHCP server configuration)
- IP address Lease time (option 51)
- Subnet mask (option 1)
- Router (gateway) (option 3)
- Others (DNS, Renewal Time value, Rebinding Time value, and so on.)

5. The DHCP client sends a DHCP request message (broadcast) with the following values:

- DA = FF:FF:FF:FF:FF:FF (broadcast)
- SA = client MAC
- SIP = 0.0.0.0
- DIP = 255.255.255.255
- Source UDP port = 68
- Destination UDP port = 67

The DHCP payload has the following values:

- Broadcast flag = 1 (broadcast) or 0 (unicast).
- Relay agent IP = 0.0.0.0
- Client MAC = mac1
- DHCP server identifier = DHCP server IP address
- Requested IP (option 50)
- Parameter request list (option 55) that lists the required items from the DHCP server to be sent along with the IP address like subnet mask, router (gateway), and others

6. The DHCP relay agent relays the DHCP Request message toward the DHCP server (unicast). The relayed packet is unicast toward the DHCP servers, with the following values:

- SIP = outgoing interface IP address by default. If source-address is configured, then the relayed packet has SIP = configured source-address.
- UDP source port = 67
- UDP destination port = 67

The DHCP payload has the following values:

- Broadcast flag = 1 (broadcast) or 0 (unicast).
- Relay agent IP = giaddr
- Client MAC = mac1
- DHCP identifier = DHCP server IP address
- Requested IP (option 50)

- Relay agent Information (option 82) if configured under dhcp-relay
- Parameter request list (option 55) that lists the required items from the DHCP server to be sent along with the IP address like subnet mask, router (gateway), and others
- Vendor specific option (if configured)

7. The DHCP server sends a DHCP Ack message to the DHCP relay agent (unicast). The DHCP Ack packet is unicasted with the following values:
   - SIP = DHCP IP address
   - DIP = giaddr
   - UDP source port = 67
   - UDP destination port = 67

   The DHCP payload has the following values:
   - Broadcast flag, either 1 (broadcast), or 0 (unicast)
   - Your (client) IP = IP address assigned by DHCP server
   - Agent IP = giaddr
   - Client MAC = mac1
   - DHCP identifier = DHCP server IP address
   - Option 82 (echoed back and based on DHCP server configuration)
   - IP address Lease time (option 51)
   - Subnet mask (option 1)
   - Router (gateway) (option 3)
   - Others (DNS, Renewal Time value, Rebinding Time value, and so on.)
8. Based on the broadcast flag sent by client, the DHCP Offer is relayed from the DHCP servers toward the client with the following values:
   - DA = FF:FF:FF:FF:FF:FF (broadcast) OR Client MAC(unicast)
   - SIP = sub-interface IP address toward the client where the DHCP relay agent is enabled
   - DIP = 255.255.255.255 (broadcast) OR Your (client) IP address (unicast)
   - Source UDP port = 67
   - Destination UDP port = 68

   The relay agent relays the DHCP Offer toward client without option 82. It strips off option 82 if echoed back from DHCP server.

   The DHCP payload has the following values:
   - Broadcast flag can be either 1 (broadcast), or 0 (unicast)
   - Your (client) IP = IP address assigned by DHCP server
   - Agent IP = giaddr
   - Client MAC = mac1
   - DHCP Server identifier (option 54) = DHCP server IP address
   - IP address lease time (option 51)

- Subnet mask (option 1)
- Router (gateway) (option 3)
- Others (DNS, Renewal Time value, Rebinding Time value, and so on.)

When renewing or releasing an address, the DHCP client unicasts the DHCP Request or Release message to the DHCP server without involvement by the DHCP relay agent.

16.1.1 Configuring DHCP relay for IPv4

Procedure

To configure DHCP relay for a subinterface:

- Configure the addresses / FQDNs of the DHCP servers.
- Optionally configure the source address for DHCP messages sent to the servers.
- Configure whether information is added to the sub-options for DHCP option 82.

Example: Configure the DHCP relay agent on a subinterface

The following example configures the DHCP relay agent on a subinterface. The example configures the IP addresses / FQDNs of the remote DHCP servers and specifies the address to be used as the GIADDR in packets sent to the servers.

The `circuit-id` and `remote-id` options are configured, which causes the DHCP relay agent to include the system_name/VRF_instance/sub-interface_id:vlan_id in the circuit-id sub-option and the DHCP client MAC address in the remote-id sub-option of DHCP option 82.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/2
 interface ethernet-1/2 {
        subinterface 1 {
            ipv4 {
                admin-state enable
                address 1.1.4.4/24 {
                }
                dhcp-relay {
                    option [
                           circuit-id
                           remote-id
                    ]
                    source-address 1.1.4.4
                    server [
                        172.16.32.1
                        172.16.64.1
                        192.168.1.1
                        remoteserver.example.com
                    ]
                }
            }
```

Example: Specify the network-instance of the DHCP server

If the DHCP server network is in a different IP-VRF network-instance from the Layer 3 subinterfaces that require DHCP relay (see Figure 7: DHCP relay using different IP-VRF or default network-instance), specify the network-instance in the configuration. For example:

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/2
 interface ethernet-1/2 {
        subinterface 1 {
            ipv4 {
                admin-state enable
                address 1.1.4.4/24 {
                }
                dhcp-relay {
                    network-instance ipvrf2
                    option [
                           circuit-id
                           remote-id
                    ]
                    source-address 1.1.4.4
                    server [
                        172.16.32.1
                        172.16.64.1
                        192.168.1.1
                        remoteserver.example.com
                    ]
                }
            }
```

16.1.2 Using the GIADDR as the source address for DHCP Discover/Request packets

Procedure

By default, the SR Linux uses the IP address of the outgoing interface as the source address for Discover/Request packets sent to the DHCP server. This is not the needed behavior for some configurations, such as a firewall protecting the DHCP server that allows connections from a limited set of IP addresses. You can use the **use-gi-addr-as-src-ip-addr** parameter to cause the SR Linux to instead use the GIADDR as the source address for Discover/Request packets sent to the DHCP server.

You can optionally configure the GIADDR address using the **gi-address** parameter. The configured GIADDR address can be a local IP address under the interface where DHCP relay is enabled, any loopback address within the same IP-VRF (if the DHCP server network is in this IP-VRF network-instance), or a loopback address defined in a different IP-VRF/default network-instance (if the DHCP server network is in different IP-VRF/default network-instance).

The following table shows the GIADDR and source address combinations.

Table 2: GIADDR and source address combinations

gi-address parameter	use-gi-addr-as-src-ipaddr parameter	GIADDR in relayed packet	Source IP address in relayed packet
Not configured (default)	False (default)	Primary IP address of interface	IP address of outgoing interface

gi-address parameter	use-gi-addr-as-src-ipaddr parameter	GIADDR in relayed packet	Source IP address in relayed packet
Configured	False (default)	Configured GIADDR	IP address of outgoing interface
Configured	True	Configured GIADDR	Configured GIADDR
Not configured (default)	True	Primary IP address of interface	Primary IP address of interface (because it is picked as the GIADDR)

Example

In the following example, the address specified with the **gi-address** parameter is used as the source address for Discover/Request packets sent to the DHCP server. If the **gi-address** parameter is not configured, then the default GIADDR (the primary IP address of the interface) is used.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/2
 interface ethernet-1/2 {
        subinterface 1 {
        ipv4 {
            admin-state enable
            address 172.16.1.1/24 {
                primary
            }
            address 172.16.2.1/24 {
            }
            dhcp-relay {
                admin-state enable
                gi-address 172.16.2.1
                use-gi-addr-as-src-ip-addr true
                option [
                    circuit-id
                    remote-id
                ]
                server [
                    1.1.1.1
                    2.2.2.2
                ]
            }
        }
```

16.1.3 Trusted and untrusted DHCP requests

If the DHCP relay agent receives a DHCP request and the downstream node added option 82 information or set the GIADDR to any value other than 0, the DHCP request is considered to be untrusted. By default, the router drops any untrusted DHCP request and discards the DHCP packets, as described in RFC 3046. SR Linux supports untrusted mode only. The DHCP relay agent discards DHCP packets traveling from the client to server side under the following conditions:

- The DHCP packet includes option 82.
- The DHCP packet has a GIADDR value that is not 0.

The DHCP relay agent discards DHCP packets traveling from the server to client side under the following conditions:

- The circuit-id or remote-id are not enabled on the relay interface, but are present in the packet.
- the GIADDR value in the DHCP packet does not match the GIADDR value on the relay interface.
- There is no matching entry in the cache.

16.2 DHCP relay for IPv6

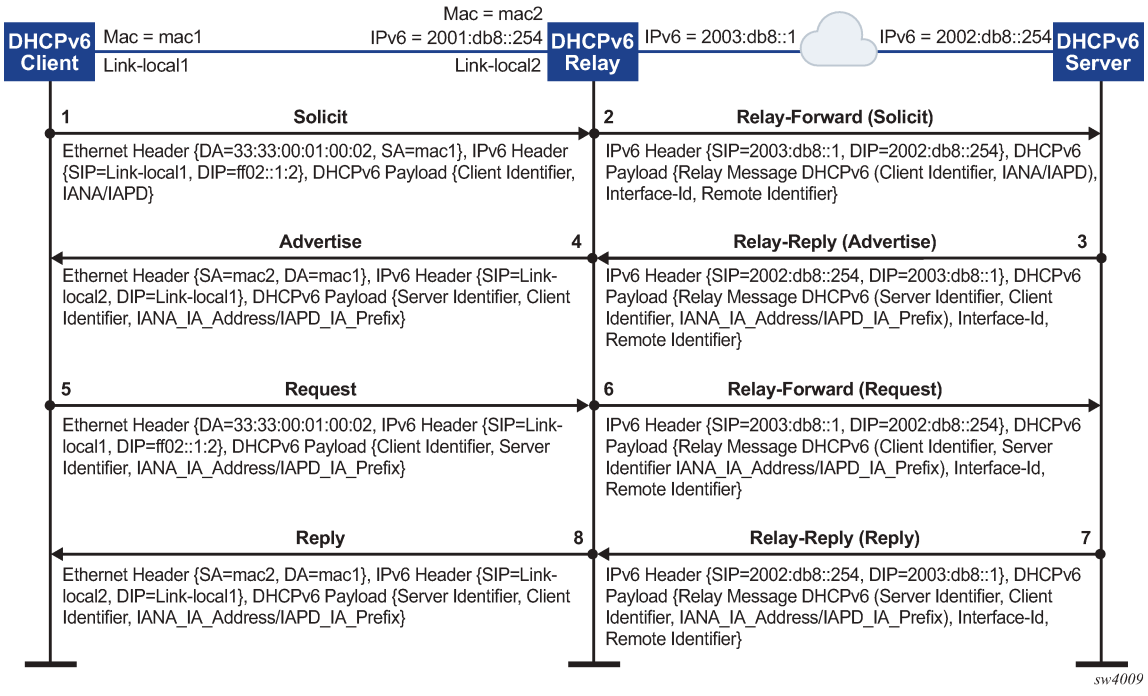
DHCP relay for IPv6 works similarly to IPv4. However, in DHCPv6, the DHCP Discover, Offer, and Ack messages are replaced by Solicit messages sent by clients, and Advertise and Reply messages sent by servers.

The DHCPv6 relay agent relays messages between clients and remote servers using Relay-Forward (client-to-server) and Relay-Reply (server-to-client) message types. DHCP option 82 is replaced in DHCPv6 by Interface-Id (option 18) and Remote Identifier (option 37), appended by relay agents.

You can optionally configure the DHCPv6 relay agent to include the client's MAC address in Client Link-Layer Address (option 79). This can be useful for dual-stack clients, where a client is using both DHCPv4 and DHCPv6, and the client's MAC address is being used as an identifier for DHCPv4.

Figure 9: DHCPv6 message flow for IPv6 address allocation shows the DHCPv6 message flow. Figure 10: DHCPv6 renew message flow and Figure 11: DHCPv6 release message flow show the renew and release flows.

Figure 9: DHCPv6 message flow for IPv6 address allocation



When assigning an address to a DHCP client, DHCP relay for IPv6 works as follows:

1. The DHCPv6 client uses its link-local address as the source IPv6 address and IPv6 multicast address FF02::1:2 and MAC address 33:33:00:01:00:02 as destination IPv6 address/MAC address respectively for solicit/request messages and with the following UDP values:
   - source UDP port = 546
   - destination UDP port = 547
2. The DHCPv6 relay agent uses a Relay-Forw message to relay the Solicit message toward the DHCPv6 server, using the outbound IPv6 address of the DHCPv6 relay agent as the source IPv6 address and with the following UDP values:
   - Source UDP port = 547
   - Destination UDP port = 547
3. The DHCPv6 server replies to the relay agent an IP address to the DHCP client, based on information in the GIADDR or in option 82, if configured to do so, and with the following UDP values:
   - Source UDP port = 547
   - Destination UDP port = 547
4. The DHCPv6 server replies to the relay agent with destination IPv6 address equal to DHCPv6 (RELAY-FW) source IPv6 address, and the following UDP values:
   - Source UDP port = 547
   - Destination UDP port = 547
5. The DHCP relay agent relays the DHCP Offer message to the DHCP client (either broadcast or unicast, based on the broadcast flag sent by the client).

Figure 10: DHCPv6 renew message flow

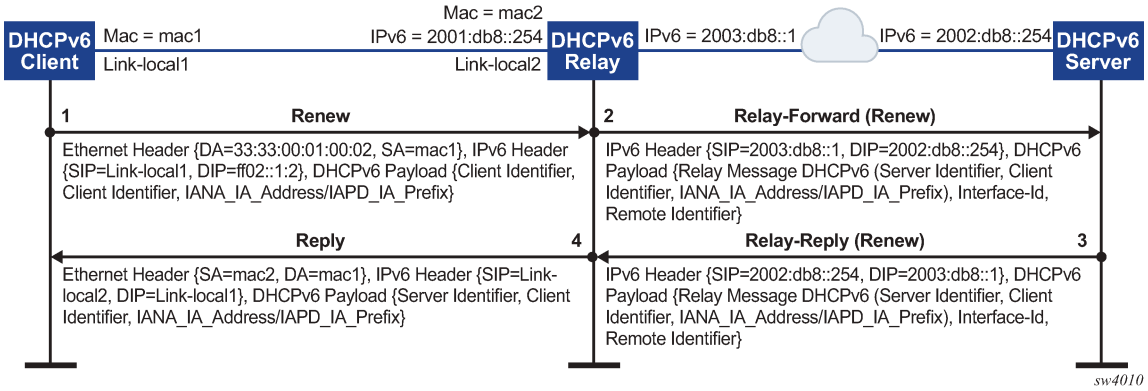
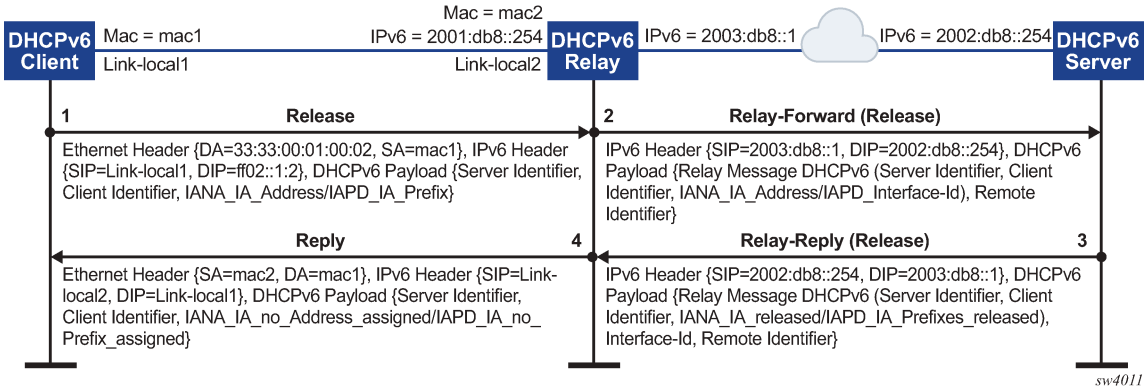


Figure 11: DHCPv6 release message flow



16.2.1 Configuring DHCP relay for IPv6

Procedure

To configure DHCP relay for a subinterface for IPv6:

- Configure the addresses / FQDNs of the DHCPv6 servers.
- Optionally configure the source IPv6 address for relay-forward messages sent to the servers.
- Optionally configure whether information is included in the Interface-Id (option 18) and Remote Identifier (option 37) in relay-forward messages.
- Optionally configure whether the MAC address of the DHCP client is included in the Client Link-Layer Address (option 79) in the relay-forward messages.

Example: Configure the DHCPv6 relay agent on a subinterface

The following example configures the DHCPv6 relay agent on a subinterface. The example configures the IP addresses / FQDNs of the remote DHCPv6 servers and specifies the address to be used as the source IPv6 address in packets sent to the servers.

The **interface-id** and **remote-id** options are configured, which causes the DHCP relay agent to include the system_name/VRF_instance/subinterface_id:vlan_id in Interface-Id (option 18) and the DHCPv6 client MAC address in the Remote Identifier (option 37).

The **client-link-layer-address** option is configured, which causes the DHCP relay agent to include the DHCPv6 client MAC address in the Client Link-Layer Address (option 79).

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/2
 interface ethernet-1/2 {
        description dut1-dut4-1
        subinterface 1 {
            ipv6 {
                admin-state enable
                address 2001:db8:101::1/64 {
                    primary
                }
                address 2001:db8:202::1/64 {
                }
                dhcp-relay {
```

```
                    admin-state enable
                    source-address 2001:db8:101::1
                    option [
                       interface-id
                       remote-id
                       client-link-layer-address
                    ]
                    server [
                       1::1
                       2::2
                       remoteserver.example.com
                    ]
                }
            }
        }
```

Example: Specify the network-instance when the DHCP server network is in a different IP-VRF

If the DHCP server network is in a different IP-VRF network-instance from the Layer 3 subinterfaces that require DHCP relay (see Figure 7: DHCP relay using different IP-VRF or default network-instance), specify the network-instance in the configuration. For example:

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/2
 interface ethernet-1/2 {
        description dut1-dut4-1
        subinterface 1 {
            ipv6 {
                admin-state enable
                address 2001:db8:101::1/64 {
                    primary
                }
                address 2001:db8:202::1/64 {
                }
                dhcp-relay {
                    network-instance ipvrf2
                    admin-state enable
                    source-address 2001:db8:101::1
                    option [
                       interface-id
                       remote-id
                       client-link-layer-address
                    ]
                    server [
                       1::1
                       2::2
                       remoteserver.example.com
                    ]
                }
            }
        }
```

16.3 QoS for DHCP relay

Self-generated DHCP/DHCPv6 packets are mapped into forwarding class 4 (fc4), low drop probability level, and DSCP marking 34 (AF41).

16.4 DHCP relay operational down reasons

The DHCP relay agent can enter an operationally down state in the following scenarios:

- The DHCP relay admin state is down.
- The subinterface under which DHCP relay is configured is operationally down.
- All DHCP servers configured within the network instance are unreachable.
- The configured GIADDR for DHCP, or source-address for DHCPv6, does not match any of the configured IP addresses under the subinterface where DHCP relay is configured
- The IP address is deleted under the subinterface.

16.5 Updating domain name resolution for DHCP-relay server FQDNs

Procedure

If the DHCP relay configuration specifies a remote DHCP server using an FQDN instead of an IP address, SR Linux periodically refreshes the state of the domain to ensure the DHCP server name can be resolved. If the name of a DHCP server cannot be resolved, the DHCP relay agent does not send requests to that DHCP server until its name can be successfully resolved.

To manually cause an update of all domain name resolutions for DHCP servers configured for DHCP relay, use the **tools system dhcp-relay update-dns-entries** command.

Example

```
--{ running }--[  ]--
# tools system dhcp-relay update-dns-entries
```

You can display the domain name resolutions with an **info from state** command. For example:

```
--{ running }--[  ]--
# info from state interface ethernet-1/1 subinterface 1 ipv4 dhcp-relay dns-resolution
    interface ethernet-1/1 {
        subinterface 1 {
            ipv4 {
                dhcp-relay {
                    dns-resolution {
                        server example.com {
                            resolved-ip-address 10.0.0.1
                            last-update "25 seconds ago"
                        }
                    }
                }
            }
        }
    }
```

16.6 Displaying DHCP relay statistics

Procedure

To display DHCP relay statistics, use the **info from state** command in candidate or running mode, or the **info** command in state mode.

Example: IPv4

```
--{ * candidate shared default }--[  ]--
# info from state interface ethernet-1/16 subinterface 1 ipv4 dhcp-relay statistics
    interface ethernet-1/16 {
        subinterface 1 {
            ipv4 {
                dhcp-relay {
                    statistics {
                        client-packets-received 2
                        client-packets-relayed 2
                        client-packets-discarded 0
                        server-packets-received 2
                        server-packets-relayed 2
                        server-packets-discarded 0
                    }
                }
            }
        }
    }
```

Example: IPv6

```
--{ * candidate shared default }--[  ]--
# info from state interface ethernet-1/16 subinterface 1 ipv6 dhcp-relay statistics
    interface ethernet-1/16 {
        subinterface 1 {
            ipv6 {
                dhcp-relay {
                    statistics {
                        client-packets-received 2
                        client-packets-relayed 2
                        client-packets-discarded 0
                        server-packets-received 2
                        server-packets-relayed 2
                        server-packets-discarded 0
                    }
                }
            }
        }
    }
```

16.6.1 Clearing DHCP relay statistics

Procedure

You can clear the DHCP relay statistics counters for a specified subinterface.

Example

```
--{ * candidate shared default }--[  ]--
```

```
# tools interface ethernet-1/2 subinterface 1 ipv4 dhcp-relay statistics clear
/interface[name=ethernet-1/2]/subinterface[index=1]:
subinterface ethernet-1/2.1 statistics cleared
```

17 DHCP server

For cases where a host requires IPAM (IP Address Management) without an external DHCP server, or where DHCP relay to underlay is not possible, IPAM information can be stored locally on the SR Linux device, which can assign an IP address and other DHCP options to the host using a local DHCP server.

SR Linux supports static IP allocations on both DHCPv4 and DHCPv6 servers. The SR Linux DHCP server can be enabled under regular Layer 3 or IRB subinterfaces. On Layer 3 or IRB subinterfaces, the DHCP server can only be enabled under subinterfaces where DHCP relay is disabled.

When an incoming DHCP Discover or Solicit message is received from a host (DHCP client), and its MAC address matches an entry in the SR Linux DHCP server configuration, the SR Linux DHCP server starts the process of IP address assignment and sends other DHCP options if configured to do so.

For DHCPv4, in addition to IP address allocation, the SR Linux can send the following DHCP options to a host:

- router (option 3) – IPv4 address of the gateway for the DHCP client
- ntp-server (option 4) – List of up to 4 NTP servers for the DHCP client to use
- dns-server (option 6) – List of up to 4 DNS servers for the DHCP client to use
- hostname (option 12) – The hostname for the DHCP client
- domain-name (option 15) – The domain name the client can use when resolving hostnames via DNS
- bootfile-name (option 66) – URL to a provisioning script for the DHCP client to use when booting
- server-id – IP address the DHCP server must match within the network-instance, such as the subinterface primary address or loopback address

For DHCPv6, in addition to IP address allocation, the SR Linux can send a list of up to 4 DNS servers for the DHCP client to use (option 23).

Notes:

- The SR Linux DHCP server supports static IP address allocation only. Dynamic allocation is not supported.
- It is assumed there is no DHCP relay agent between the DHCP client and the SR Linux DHCP server. Relayed frames are not supported.
- For IPv6, DHCP configuration uses MAC-to-IPv6 address binding. The IPv6 address is assigned to the client based on the client's MAC address, not IAID. The client's MAC address is derived from the client identifier. The recommended client identifier type is DUID type DUID-LLT or DUID-LL.

17.1 Configuring the DHCP server

Procedure

To configure the SR Linux DHCP server, you enable it on a subinterface and at the `system dhcp-server` level configure DHCPv4 and DHCPv6 options and static IP allocations for the network-instance where DHCP is required.

Example: Enable DHCPv4 and DHCPv6 servers for a subinterface

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1 subinterface 1
    interface ethernet-1/1 {
        subinterface 1 {
            admin-state enable
            ipv4 {
                admin-state enable
                address 192.14.1.4/27 {
                }
                dhcp-server {
                    admin-state enable
                }
            }
            ipv6 {
                admin-state enable
                address 2001:192:14:1::4/120 {
                }
                dhcpv6-server {
                    admin-state enable
                }
            }
        }
    }
```

Example: Configure DHCPv4 and DHCPv6 options

The following example configures DHCPv4 and DHCPv6 options, which are supplied to DHCP clients on the default network-instance:

```
--{ * candidate shared default }--[  ]--
# info system dhcp-server
    system {
        dhcp-server {
            admin-state enable
            network-instance default {
                dhcpv4 {
                    options {
                        domain-name lan
                        router 192.168.1.1
                        dns-server [
                            192.168.1.53
                            192.168.1.54
                        ]
                        ntp-server [
                            192.168.1.50
                        ]
                    }
                }
                dhcpv6 {
                    options {
                        dns-server [
                            2001:192:14:1::4
                            2001:192:14:1::5
                        ]
                    }
                }
            }
        }
    }
```

Example: Configure IPv4 static IP allocation settings

The following example configures static IP allocation settings for an IPv4 host:

```
--{ * candidate shared default }--[  ]--
# info detail system dhcp-server network-instance default dhcpv4
    system {
        dhcp-server {
            admin-state enable
            network-instance default {
                dhcpv4 {
                    admin-state enable
                    static-allocation {
                        host 00:1D:FE:E0:E9:7C {
                            ip-address 192.168.1.1/24
                            options {
                                router 192.168.1.1
                                dns-server [
                                    192.168.1.53
                                ]
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Configure IPv6 static IP allocation settings

The following example configures static IP allocation settings for an IPv6 host:

```
--{ * candidate shared default }--[  ]--
# info detail system dhcp-server network-instance default dhcpv6
    system {
        dhcp-server {
            admin-state enable
            network-instance default {
                dhcpv6 {
                    admin-state enable
                    static-allocation {
                        host 92:93:47:30:32:CA {
                            ip-address 2001:1::192:168:12:1/126
                            options {
                                dns-server [
                                    2001:192:14:1::4
                                ]
                            }
                        }
                    }
                }
            }
        }
    }
```

18 IPv6 router advertisements

You can configure an IPv6 subinterface to originate Router Advertisement (RA) messages. The following settings can be configured for the RA messages:

- Current hop limit to advertise in the RA messages.
- IP MTU. Hosts can associate the IP MTU with the link on which the RA messages are received.
- Managed configuration flag. When enabled, this setting indicates that hosts should use DHCPv6 to obtain IPv6 addresses.
- Other configuration flag. When enabled, this setting indicates that hosts should use DHCPv6 to obtain other configuration information (besides addresses).
- The maximum and minimum time between sending router advertisement messages to the all-nodes multicast address.
- IPv6 prefix list. Hosts that support Stateless Address Auto-Configuration (SLAAC) can use the IPv6 prefixes in the RA messages to generate IPv6 addresses.
- Number of milliseconds advertised for the reachable time and the retransmit time in RA messages, which hosts use for address resolution and the ICMPv6 Neighbor Unreachability Detection algorithm.
- Number of seconds advertised as the router lifetime in RA messages. This setting indicates amount of time the advertising router can be used as a default router/gateway.

18.1 Configuring IPv6 router advertisements

Procedure

You can configure SR Linux to originate RA messages from an IPv6 subinterface. The RA messages include an IPv6 prefix that SLAAC-enabled clients can use to generate IPv6 addresses.

Example

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1 subinterface 1 ipv6 router-advertisement
    interface ethernet-1/1 {
        subinterface 1 {
            ipv6 {
                admin-state enable
                router-advertisement {
                    router-role {
                        admin-state enable
                        prefix 2001:db8:0:b::/64 {
                        }
                    }
                }
            }
        }
    }
```

19 IPv6 Router Advertisement guard (RA guard)

IPv6 Router Advertisement guard (IPv6 RA guard) allows you to configure policies that filter out IPv6 RA messages that may be incorrectly or maliciously configured. IPv6 RA messages entering a subinterface where an IPv6 RA guard policy is applied can be accepted or discarded based on match criteria specified in the policy.

IPv6 RA guard is supported on Layer 2 and Layer 3 subinterfaces, which allows unwanted RA messages to be discarded as close to the network edge or server connection as possible. The IPv6 RA guard feature can be configured on 7220 IXR-D1, D2, and D3 systems only.

On IRB interfaces, an IPv6 RA guard policy can be applied to the Layer 2 subinterface, but not on the IRB subinterface.

Ingress ACLs are applied before IPv6 RA guard policies, which may cause RA messages to be discarded before they can be evaluated by an IPv6 RA guard policy

Note:
Ingress ACLs are applied before IPv6 RA guard policies, which may cause RA messages to be discarded before they can be evaluated by an IPv6 RA guard policy

The following can be used as match criteria in an IPv6 RA guard policy:

- Advertised IPv6 prefix set
- Source IPv6 address list or prefix set
- RA hop-count limit
- Router preference value
- Managed configuration flag (M-flag) setting
- Other configuration flag (O-flag) setting

An IPv6 RA guard policy can have an action of accept or discard. When an IPv6 RA guard policy is applied to a subinterface, the default action for the subinterface is the opposite of the action specified in the policy. If the policy action is accept, then IPv6 RA packets that do not match the policy are discarded; if the policy action is discard, IPv6 RA packets that do not match the policy are accepted.

To configure IPv6 RA guard, you specify match criteria and an action in an IPv6 RA guard policy, then apply the policy to a subinterface. If an IPv6 RA guard policy is not applied to a subinterface, then IPv6 RA guard is disabled on that subinterface.

Note:
Depending on your configuration, it may be more efficient to block IPv6 RA messages on a subinterface using an ACL entry and action, instead of configuring an IPv6 RA guard policy.

19.1 Configuring IPv6 RA guard policies

Procedure

To configure an IPv6 RA guard policy, specify one or more match criteria and an action of either accept or discard.

Example: Configure an IPv6 RA guard policy with accept action

The following example configures an IPv6 RA guard policy with an advertised IPv6 prefix set and source IPv6 prefix set as match criteria, and accept as the action.

To be considered a match, all advertised prefixes in the RA message must match the IPv6 prefix set, and the source address of the RA message must match the source IPv6 address prefix set.

```
--{ * candidate shared default }--[  ]--
# info system ra-guard-policy
    system {
        ra-guard-policy rag1 {
            action accept
            advertise-prefix-set 2001:db8:0:b::/64
            source-prefix-set 2001:1::192:168:11:1/126
        }
    }
```

Example: Configure an IPv6 RA guard policy with discard action

The following example configures an IPv6 RA guard policy with no match criteria and action of discard. This policy blocks all RA messages on subinterfaces where it is applied.

```
--{ * candidate shared default }--[  ]--
# info system ra-guard-policy
    system {
        ra-guard-policy "Discard all" {
            action discard
        }
    }
```

19.2 Applying IPv6 RA guard policies to subinterfaces

Procedure

To activate IPv6 RA guard, apply an IPv6 RA guard policy to a subinterface.

Example: Apply an IPv6 RA guard policy to a subinterface

The following example applies an IPv6 RA guard policy to a subinterface. This policy (configured in the previous example) causes all IPv6 RA messages received on the subinterface to be discarded.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/4 subinterface 2 ra-guard
    interface ethernet-1/4 {
        subinterface 2 {
            ra-guard {
                policy "Discard all"
            }
        }
```

```
    }
```

If the subinterface has VLANs configured, you can specify a list of VLANs to which the IPv6 RA guard policy applies. If a VLAN list is specified, the IPv6 RA guard policy applies only to those VLANs, not to any others configured on the subinterface. If VLAN list is not specified, the policy applies to all VLANs on the subinterface.

Example: Specify a VLAN list with the IPv6 RA guard policy

On a default bridged subinterface, where the `vlan encap single-tagged vlan-id any` setting is configured, a VLAN list must be specified with the IPv6 RA guard policy. For example:

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/4 subinterface 2
    interface ethernet-1/4 {
        subinterface 2 {
            admin-state enable
            type bridged
            vlan {
                encap {
                    single-tagged {
                        vlan-id any
                        }
                    }
                }
            }
            ra-guard {
                policy rag1
                vlan-list 10 {
                }
            }
        }
    }
```

20 Interface port speed configuration

By default, ports on SR Linux devices operate at the speeds listed in Table 3: Default and supported port speeds for SR Linux devices. You can optionally configure ports to operate a different speed as long as that speed is supported for the port. On all devices, the Mgmt ports operate at 1G.

On 7220 IXR-D1 systems, it is possible to configure auto-negotiation for the port. See Configuring link auto-negotiation (7220 IXR-D1 only).

Table 3: Default and supported port speeds for SR Linux devices

SR Linux Device	Port range	Default port speed	Supported port speeds
7250 IXR	IMM ports 1-32	100G	40G, 100G **Note:** Ports 9-12 cannot operate at different port speeds (some at 40G and others at 100G). The required speed of ports 9-12 is based on the port-speed of the lowest-numbered configured port in this block; if any higher-numbered port in the block is configured with a different port speed that port does not come up.
	IMM ports 33-36	100G	40G, 100G, 400G
7220 IXR-D1	Ports 1-48	1G	10M, 100M, 1G
	Ports 49-52	10G	10G
7220 IXR-D2	Ports 1-48	25G	1G, 10G, 25G **Note:** If one port in each consecutive group of 4 ports (1-4, 5-8, and so on) is 25G, the other 3 ports must also be 25G. If one port in each consecutive group of 4 ports is 1G or 10G, the other 3 ports must also be 1G or 10G.
	Ports 49-56	100G	40G, 100G
7220 IXR-D3	Ports 1-2	10G	10G
	ethernet-1/[3-34]	100G	40G, 100G
	ethernet-1/[3-33]/n	none	10G, 25G
7220 IXR-H2	Ports 1-128	100G	100G
7220 IXR-H3	Ports 1-2	10G	10G

SR Linux Device	Port range	Default port speed	Supported port speeds
	Ports 3-34	400G	40G, 100G, 400G

20.1 Configuring interface port speed

Procedure

You can configure the port speed for an interface.

Example

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1 ethernet
    interface ethernet-1/1 {
        ethernet {
            port-speed 100G
            }
        }
    }
```

20.2 Configuring link auto-negotiation (7220 IXR-D1 only)

Procedure

For ports 1-48 on 7220 IXR-D1 systems, you can configure the interface to use auto-negotiation for the speed, duplex, and flow-control settings. Table 4: Port speed negotiation for RJ-45 ports (7220 IXR-D1 systems) lists how the auto-negotiation setting inter-operates with the port-speed configured for the interface.

Table 4: Port speed negotiation for RJ-45 ports (7220 IXR-D1 systems)

auto-negotiate parameter setting	Port speed parameter configured?	Port speed behavior
false	No	Bring up the port at the default speed of 1G.
false	Yes	Bring up the port at the configured speed of 10M, 100M or 1G if the other side is configured for the same speed; otherwise, the port state is oper-down.
true (default)	No	All speeds are advertised as supported, and the actual speed is the highest common value between the two sides.
true (default)	Yes	The configured port speed is the only speed advertised. If the port at the other side of the link

auto-negotiate parameter setting	Port speed parameter configured?	Port speed behavior
		advertises this speed as well, the link comes up; otherwise it stays down.

Example

The following example configures auto-negotiation and port speed for a port on a 7220 IXR-D1 system. In this example, a port speed is configured, and the auto-negotiation setting is enabled. The SR Linux advertises the configured port speed to the other end of the link. If the other port also advertises this speed, then the link is established; otherwise, the port state is oper-down.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1 ethernet
    interface ethernet-1/1 {
        ethernet {
            auto-negotiate true
            port-speed 100M
            }
        }
    }
```

21 Interface hold-timers

You can configure hold-timers that keep an interface operationally enabled or disabled for a specified amount of time following an event that brings the interface up or shuts the interface down.

For example, you can configure a hold-timer that keeps an interface operationally disabled for a period of time following a system reboot, and you can configure a hold-timer that keeps an interface operationally enabled for a period of time after the interface goes down.

The main use for hold-timers is to reduce the number of link transitions and advertise/withdraw messages in networks where there are flapping optics.

You can configure a hold-time up timer and a hold-time down timer for an interface:

- hold-time up timer

  This timer specifies the amount of time an interface is kept operationally disabled following an event that would normally enable it, such as entering the **interface admin-state enable** command or a system reboot.

  The interface remains disabled from the time the event occurs until the hold-time up timer expires. While the hold-time up timer is running, the transceiver is enabled but the system does not consider the interface operationally up until the timer expires.

- hold-time down timer

  This timer specifies the amount of time an interface remains operationally enabled following an event that brings the interface down. When triggered, the hold-time down timer keeps the interface operationally enabled until the timer expires. Entering the **interface admin-state disable** command does not trigger the hold-time down timer, nor does internal events such as fabric unavailability.

  If you manually disable the interface while the hold-time down timer is running, the interface is disabled immediately, and the timer is aborted.

The hold-timers can be set to a value from 1-86,400 seconds. There is no default value; if not configured, no hold-time is considered when an interface changes state.

The hold-timers are available for Ethernet interfaces only, including those that are part of a LAG. You cannot configure a hold-timer for an interface in breakout mode.

The hold-timer does not affect the port LED color, which reflects the physical status of the port; that is, the port LED is green when the hold-time up timer is running, and solid amber when the hold-time down timer is running.

21.1 Configuring interface hold timers

Procedure

To configure hold-timers for an interface, specify a time value from 1-86,400 seconds for the hold-time up and, or hold-time down timers.

Example: Configure a hold-time up and a hold-time down timer for an interface

In the following example, when the interface is enabled, it remains operationally disabled for 200 seconds, until the hold-time up timer expires. When the interface becomes disabled, it remains operationally enabled for 100 seconds, until the hold-time down timer expires.

```
--{ candidate shared default }--[  ]--
# info interface ethernet-1/1 ethernet
    interface ethernet-1/1 {
        ethernet {
            hold-time {
                up 200
                down 100
            }
        }
    }
```

Example: Display the amount of time remaining for the hold-timer

When a hold-timer is in effect, you can use the **info from state** command to display the amount of time remaining. For example:

```
--{ running }--[  ]--
# info from state interface ethernet-1/1 ethernet hold-time
    interface ethernet-1/1 {
        ethernet {
            hold-time {
                up 200
                down 100
                up-expires 85 seconds from now
            }
        }
    }
```

In addition, when the hold-time up timer is in effect, the `port-oper-down-reason` for the interface is shown as `interface-hold-time-up-active`.

22 Reload-delay timer

After the system boots, the reload-delay timer keeps an interface shut down with the laser off for a configured amount of time until connectivity with the rest of network is established. When applied to an access multi-homed interface (typically an Ethernet Segment interface), this delay can prevent black-holing traffic coming from the multi-homed server or CE.

When a reload-delay timer is configured, the interface port is shut down and the laser is turned off from the time that the system determines the interface state following a reboot or reload of the XDP process, until the number of seconds specified in the reload-delay timer elapse.

The reload-delay timer is only supported on Ethernet interfaces that are not enabled with breakout mode. For a multi-homed LAG interface, the reload-delay timer should be configured on all the interface members. The reload-delay timer can be from 1-86,400 seconds. There is no default value; if not configured for an interface, there is no reload-delay timer.

Only ES interfaces should be configured with a non-zero reload-delay timer. Single-homed interfaces and network interfaces (used to forward VXLAN traffic) should not have a reload-delay timer configured.

The following example sets the reload-delay timer for an interface to 20 seconds. The timer starts following a system reboot or when the IMM is reconnected, and the system determines the interface state. During the timer period, the interface is deactivated and the port laser is inactive.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1
    interface ethernet-1/1 {
        admin-state enable
        ethernet {
            reload-delay 20
        }
    }
```

When the reload-delay timer is running, the `port-oper-down-reason` for the port is shown as `interface-reload-timer-active`. The `reload-delay-expires` state indicates the amount of time remaining until the port becomes active. For example:

```
--{ running }--[  ]--
# info from state interface ethernet-1/1
    interface ethernet-1/1 {
        description eth_seg_1
        admin-state enable
        mtu 9232
        loopback-mode false
        ifindex 671742
        oper-state down
        oper-down-reason interface-reload-time-active
        last-change "51 seconds ago"
        linecard 1
        forwarding-complex 0
        vlan-tagging true
        ...
        ethernet {
            auto-negotiate false
            lacp-port-priority 32768
            port-speed 100G
```

```
            hw-mac-address 00:01:01:FF:00:15
            reload-delay 20
            reload-delay-expires "18 seconds from now"
            flow-control {
                receive false
                transmit false
            }
        }
    }
```

22.1 Configuring the reload-delay timer for an interface

Procedure

To configure the reload-delay timer for an interface, you specify a timer value from 1-86,400 seconds. The timer starts following a system reboot or when the IMM is reconnected, and the system determines the interface state. During the timer period, the interface is deactivated and the port laser is inactive. You can display information about an active reload-delay timer by entering the **info from state** command for the interface.

Example: Set the reload-delay timer for an interface

The following example sets the reload-delay timer for an interface to 20 seconds.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1
    interface ethernet-1/1 {
        ethernet {
            reload-delay 20
        }
    }
```

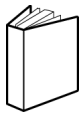
Example: Display reload-delay information

When the reload-delay timer is running, the `port-oper-down-reason` for the port is shown as `interface-reload-timer-active`. The `reload-delay-expires` state indicates the amount of time remaining until the port becomes active. For example:

```
--{ running }--[  ]--
# info from state interface ethernet-1/1
    interface ethernet-1/1 {
        description eth_seg_1
        admin-state enable
        mtu 9232
        loopback-mode false
        ifindex 671742
        oper-state down
        oper-down-reason interface-reload-time-active
        last-change "51 seconds ago"
        linecard 1
        forwarding-complex 0
        vlan-tagging true
        ...
        ethernet {
            auto-negotiate false
            lacp-port-priority 32768
            port-speed 100G
            hw-mac-address 00:01:01:FF:00:15
```

```
            reload-delay 20
            reload-delay-expires "18 seconds from now"
            flow-control {
                receive false
                transmit false
            }
        }
    }
```

Customer document and product support



Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback