NOKIA

Nokia Service Router Linux

CONFIGURATION BASICS
RELEASE 21.11

3HE 17901 AAAA TQZZA
Issue 1

December 2021

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 basic configuration for 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.

1.1 What's new

Topic	Location
Configurable hash options for IP ECMP load balancing	IP ECMP Load Balancing
Authorization for individual service types	Service authorization for local users
DHCP server with static allocation	DHCP server
DHCP relay using different IP-VRF or default network-instance	DHCP relay
Authorization using TACACS+ priv-lvl mapping	Authorization using a TACACS+ server
eBGP multihop	eBGP multihop
Upgrading the configuration following a software upgrade	Configuration upgrades

1.2 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.

1.3 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**.
- Angle brackets (< >) indicate an item that is not used verbatim. For example, for the command **show ethernet <***name***>**, *name* should be replaced with the name of the interface.
- 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 System management

This chapter contains procedures for setting up basic system management functions on SR Linux, including the hostname, domain name, DNS settings, and management network-instance. It contains examples of configuring an SSH server and FTP server, as well as NTP for the system clock, and enabling an SNMP server.

2.1 Configuring a hostname

The SR Linux device must have a hostname configured. The default hostname is `srlinux`. The hostname normally appears on all CLI prompts on the device, although you can override this with the **environment prompt** CLI command.

The hostname should be a unique name on the network, and can be a fully qualified domain name (FQDN), or an unqualified single-label name. If the hostname is a single-label name (for example, `srlinux`), the system may use its domain name, if configured, to infer its own FQDN.

Example:

The following example shows the configuration for a hostname on the SR Linux device.

```
--{ candidate shared default }--[  ]--
# info system name
    system {
        name {
            host-name 3-node_srlinux-A
        }
    }
```

2.2 Configuring a domain name

The SR Linux device uses its hostname, combined with a domain name to form its fully qualified domain name (FQDN). It is expected that the FQDN exists within the DNS servers used by SR Linux, though this is not a requirement.

Assuming the SR Linux FQDN is in the DNS server, you can use the FQDN to reach the SR Linux device without knowing its management address. A domain name is not mandatory, but if specified, it is added to the DNS search list by default.

Example:

The following shows the configuration for a domain name on the SR Linux device. In this example, the device FQDN is set to `3-node_srlinux-A.mv.usa.nokia.com`.

```
--{ candidate shared default }--[  ]--
# info system name
    system {
        name {
            host-name 3-node_srlinux-A
            domain-name mv.usa.nokia.com
```

```
        }
    }
```

2.3 Configuring DNS settings

The SR Linux device uses DNS to resolve hostnames within the configuration, or for operational commands, such as ping. You can specify up to three DNS servers for the SR Linux device to use, with either IPv4 or IPv6 addressing.

You can also specify a search list of DNS suffixes that the device can use to resolve single-label names; for example, for a search list of `nokia1.com` and `nokia2.com`, a ping for host `srlinux` does a DNS lookup for `srlinux.nokia1.com`, and if unsuccessful, does a DNS lookup for `srlinux.nokia2.com`.

The SR Linux device supports configuration of static DNS entries. Static DNS entries allow resolution of hostnames that may not be in the DNS servers used by the SR Linux device. Using a static DNS entry, you can map multiple addresses (both IPv4 and IPv6) to one hostname. The SR Linux linux_mgr application adds the static DNS entries to the `/etc/hosts` file in the underlying Linux OS.

Example:

In the following example, the SR Linux device is configured to use two DNS servers to resolve hostnames, a search list of DNS suffixes for resolving single-label names, and IPv4 and IPv6 static DNS entries for a host.

DNS requests are sourced from the mgmt network-instance (see Configuring the management network-instance).

```
--{ candidate shared default }--[  ]--
# info system dns
    system {
        dns {
            network-instance mgmt
            server-list [
                1.1.1.1
                2.2.2.2
            ]
            search-list [
                nokia1.com
                nokia2.com
            ]
            host-entry srlinux.nokia.com {
                ipv4-address 3.3.3.3
                ipv6-address 2001:db8:123:456::11:11
            }
        }
    }
```

2.4 Configuring the management network-instance

Management of the SR Linux device is primarily done via a management network-instance. The management network-instance isolates management traffic from other network-instances configured on the device.

The out-of-band `mgmt0` interface is automatically added to the management network-instance, and management services run within the management network-instance.

Although the management network-instance is primarily intended to handle management traffic, you can configure it in the same way as any other network-instance on the device, including protocols, policies, and filters. The management network instance is part of the default configuration, but may be deleted if necessary.

Addressing within the management network-instance is available via DHCP and static IP addresses. Both IPv4 and IPv6 addresses are supported.

Example:

```
--{ candidate shared default }--[  ]--
# info network-instance mgmt
    network-instance mgmt {
        type ip-vrf
        admin-state enable
        description "Management network instance"
        interface mgmt0.0 {
        }
        protocols {
            linux {
                export-routes true
                export-neighbors true
            }
        }
    }
```

2.5 Access types

Access to the SR Linux device is available via a number of APIs and protocols. The SR Linux supports the following ways to access the device:

- SSH – Secure Shell, a standard method for accessing network devices. See Enabling an SSH server.
- FTP – File Transfer Protocol, a secure method for ing files to and from network devices. See Configuring FTP.
- Console – Access to the SR Linux CLI via direct connection to a serial port on the device.
- gNMI – A gRPC-based protocol for the modification and retrieval of configuration from a target device, as well as the control and generation of telemetry streams from a target device to a data collection system. See gNMI server.
- JSON-RPC – Ability to retrieve and set configuration and state using a JSON-RPC API. See JSON-RPC server.
- SNMP – Simple Network Management Protocol, a commonly used network management protocol. The SR Linux device supports SNMPv2 with a limited set of OIDs.

Regardless of the method of access, all sessions are authenticated (if authentication is enabled), whether the session is entered via the console, SSH, or an API. Access to the device is controlled via the aaa_mgr application. See Securing access.

2.5.1 Enabling an SSH server

You can enable an SSH server for one or more network instances on the SR Linux device, so that users can log in to the CLI using an SSH client. The SR Linux device implements SSH via OpenSSH, and configures `/etc/ssh/sshd_config` in the underlying Linux OS. Only SSHv2 is supported.

Example:

In the following example, an SSH server is enabled in the mgmt and default network-instances, specifying the IP addresses where the device listens for SSH connections:

```
--{ candidate shared default }--[  ]--
# info system ssh-server
    system {
        ssh-server {
            network-instance mgmt {
                admin-state enable
                source-address [
                    1.1.1.1
                    1.1.1.2
                ]
            }
            network-instance default {
                admin-state enable
                source-address [
                    2.1.1.1
                    2.1.1.2
                ]
            }
        }
    }
```

2.5.1.1 Configure SSH key-based authentication

The SR Linux SSH server supports RSA public-private key-based authentication, where an SSH client provides a signed message that has been encrypted by a private key. If the SSH client's corresponding public key is configured on the SR Linux, the SSH server can authenticate the client.

When performing authentication for a user, the SR Linux first tries public-key authentication; if this fails, the SR Linux tries password authentication.

To configure SSH key-based authentication, you generate a public-private key pair, then add the public key to the SR Linux.

Example:

The following is an example of using the ssh-keygen utility in Linux to generate an RSA key pair with a length of 2,048 bits:

```
# ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:RNVV8/XRVK7PhY2OJxa7rjkSUFqyVoj4pUXL2PDs7mI user@linux
The key's randomart image is:
+---[RSA 2048]----+
| .o+o. ...oo*|
| o.oB*.. +=|
| . .o@* +|
| Fo = |
| . .M . + o|
| .. = o.|
| .. = o o|
```

```
| E.. .o + |
| . ...o+o |
+----[SHA256]-----+
```

Example

After generating the RSA key pair, you can add the public key to the SR Linux. The location for the public key depends on the type of user for which SSH key-based authentication is being configured:

- For Linux users (see Linux users), you add the public key to the user's `$HOME/.ssh/authorized_keys` file.
- For users configured within the SR Linux CLI (see Local users), you add the public key to the SR Linux configuration file. This can be done with a CLI command.

For example, the following CLI command configures a public key and password for the SR Linux user `srlinux`:

```
--{ candidate shared default }--[  ]--
# system aaa authentication user username srlinux ssh-key
[ <public-key> ] password <password>
```

In the example, the `<public-key>` has the format `ssh-rsa <key> <comment>`. If multiple public keys are configured for a user, they are tried in the order they were configured.

2.5.2 Configuring FTP

You can enable an FTP server for one or more network instances on the SR Linux device, so that users can transfer files to and from the device. The SR Linux uses the vsftpd (very secure FTP daemon) application within the underlying Linux OS. The authenticated user's home directory returned by the aaa_mgr application is set as the user's FTP root directory.

Example:

In the following example, the FTP server is enabled in the mgmt and default network-instance, specifying the IP addresses where the device listens for FTP connections:

```
--{ candidate shared default }--[  ]--
# info system ftp-server
    system {
        ftp-server {
            network-instance mgmt {
                admin-state enable
                source-address [
                    1.1.1.1
                ]
            }
            network-instance default {
                admin-state enable
                source-address [
                    2.1.2.1
                ]
            }
        }
    }
```

2.6 Configuring banners

You can specify banner text that appears when a user connects to the SR Linux device. The following banners can be configured:

- Login banner – Displayed before a user has been authenticated by the system (for example, at the SSH login prompt)
- Message of the day (motd) banner – Displayed after the user has been authenticated by the system

The banners appear regardless of the method used to connect to the SR Linux, so they are displayed to users connecting via SSH, console, and so on.

Example:

In the following example, login and motd banners are configured. The login banner text appears at the prompt when a user attempts to log in to the system, and the motd banner text appears after the user has been authenticated.

```
--{ candidate shared default }--[  ]--
# info system banner
    system {
        banner {
            login-banner "Enter your SRLinux login credentials."
            motd-
banner "Welcome to the SRLinux CLI. Note that your activity may be monitored."
        }
    }
```

2.7 Synchronizing the system clock

Network Time Protocol (NTP) is used to synchronize the system clock to a time reference. You can configure NTP settings on the SR Linux device using the CLI, and the SR Linux linux_mgr application provisions the settings in the underlying Linux OS.

NTP does not account for time zones, instead relying on the host to perform such computations. Time zones on the SR Linux device are based on the IANA tz database, which is implemented by the underlying Linux OS. You can specify the time zone of the SR Linux device using the CLI.

Example:

The following configuration enables the system NTP client on the SR Linux device and specifies an NTP server to use for clock synchronization. The NTP client runs in the mgmt network-instance. The system time zone is set to America/Los_Angeles.

```
--{ candidate shared default }--[  ]--
# info system ntp
    system {
        ntp {
            admin-state enable
            network-instance mgmt
            server 4.53.160.75 {
            }
        }
        clock {
            timezone America/Los_Angeles
        }
    }
```

2.8 Configuring SNMP

The SR Linux device supports SNMPv2. See the *SR Linux System Management Guide* for descriptions of the supported SNMP OIDs. The MIB file that covers these OIDs is packaged with each release.

Example:

In the following example, an SNMP server is running within the mgmt and default network-instances, and the configuration specifies the IP addresses where the device listens for SNMP client connections:

```
--{ candidate shared default }--[  ]--
# info system snmp
    system {
        snmp {
            community test1 {
                permission r
                version v2c
            }
            network-instance mgmt {
                admin-state enable
                source-address [
                    1.1.1.1
                ]
            }
            network-instance default {
                admin-state enable
                source-address [
                    3.3.3.3
                ]
            }
        }
    }
```

2.9 IP ECMP Load Balancing

Equal-Cost Multipath Protocol (ECMP) refers to the distribution of packets over two or more outgoing links that share the same routing cost. Static, IS-IS, OSPF, and BGP routes to IPv4 and IPv6 destinations can be programmed into the datapath by their respective applications, with multiple IP ECMP next-hops.

The SR Linux load-balances traffic over multiple equal-cost links with a hashing algorithm that uses header fields from incoming packets to calculate which link to use. When an IPv4 or IPv6 packet is received on a subinterface, and it matches a route with a number of IP ECMP next-hops, the next-hop that forwards the packet is selected based on a computation using this hashing algorithm. The goal of the hash computation is to keep packets in the same flow on the same network path, while distributing traffic proportionally across the ECMP next-hops, so that each of the *N* ECMP next-hops carries approximately 1/*N*th of the load.

The hash computation takes various key and packet header field values as inputs and returns a value that indicates the next-hop. The key and field values that can be used by the hash computation depend on the platform, packet type, and configuration options, as follows:

On 7250 IXR systems, the following can be used in the hash computation:

- For IPv4 TCP/UDP non-fragmented packets: user-configured hash-seed (0-65535; default 0), source IPv4 address, destination IPv4 address, IP protocol, L4 source port, L4 destination port. The algorithm is asymmetric; that is, inverting source and destination pairs does not produce the same result.

- For IPv6 TCP/UDP non-fragmented packets: user-configured hash-seed (0-65535; default 0), source IPv6 address, destination IPv6 address, IPv6 flow label (even if it is 0), IP protocol (IPv6 next-header value in the last extension header), L4 source port, L4 destination port. The algorithm is symmetric; that is, inverting source and destination pairs produces the same result.
- For all other packets: user-configured hash-seed (0-65535; default 0), source IPv4 or IPv6 address, destination IPv4 or IPv6 address.

On 7220 IXR-D1, D2, D3 and 7220 IXR-H2 and H3 systems, the following can be used in the hash computation:

- For IPv4 TCP/UDP non-fragmented packets: VLAN ID, user-configured hash-seed (0-65535; default 0), source IPv4 address, destination IPv4 address, IP protocol, L4 source port, L4 destination port. The algorithm is asymmetric.
- For IPv6 TCP/UDP non-fragmented packets: VLAN ID, user-configured hash-seed (0-65535; default 0), source IPv6 address, destination IPv6 address, IPv6 flow label (even if it is 0), IP protocol (IPv6 next-header value in the last extension header), L4 source port, L4 destination port.
- For all other packets: user-configured hash-seed (0-65535; default 0), source IPv4 or IPv6 address, destination IPv4 or IPv6 address.

2.9.1 Configuring IP ECMP load balancing

To configure IP ECMP load balancing, you specify hash-options that are used as input fields for the hash calculation, which determines the next-hop for packets matching routes with multiple ECMP hops.

Example:

The following example configures hash options for IP ECMP load balancing, including a hash seed and packet header field values to be used in the hash computation.

```
--{ * candidate shared default }--[  ]--
# info system load-balancing
    system {
        load-balancing {
            hash-options {
                hash-seed 128
                ipv6-flow-label false
            }
        }
    }
```

If no value is configured for the **hash-seed** the default value is 0. If a hash-option is not specifically configured, the default is **true**.

On 7250 IXR systems, if **source-address** is configured as a hash option, the **destination-address** must also be configured as a hash option. Similarly, if **source-port** is configured as a hash option, the **destination-port** must also be configured as a hash option.

3 Configuration management

The chapter describes concepts behind managing the SR Linux configuration, including configuration datastores, modes, and how to commit changes to the running configuration.

3.1 Default configuration

At startup, the SR Linux loads a JSON configuration file, located at `/etc/opt/srlinux/config.json`. If this startup configuration does not exist, the system is started using a factory default configuration.

The factory default configuration brings device into management, enables DHCP/v6 on the management interface, adds it to the management network-instance, enables an SSH server, and creates various system logs and applies a default set of CPM filters.

You can optionally create a rescue configuration, which is loaded if the startup configuration fails to load (see Rescue configuration). If the startup configuration fails to load, and no rescue configuration exists, the system is started using the factory default configuration.

3.2 Configuration datastores

Configuration and state information reside in datastores on the SR Linux device. The following datastores are available:

- Running – contains the currently active configuration.
- State – contains the running configuration, plus dynamically added data such as operational state of interfaces or BGP peers added via auto-discovery, as well as session states and routing tables.
- Candidate – contains a user-configurable version of the running datastore. After it has been committed, the candidate datastore becomes the running datastore.
- Tools – contains executable commands that allow you to perform operations such as restarting the device and clearing interface statistics.

Within the CLI, you can use the **info** command to display information from a datastore. For example, entering the **info from state** command (or entering the **info** command in state mode) displays configuration and statistics from the state datastore for the current context, and entering the **info from running** command (or the **info** command in running mode) displays configuration from the running datastore for the current context.

3.3 Configuration modes

The candidate datastore corresponds to a configuration mode within the SR Linux CLI. In candidate mode, you can modify SR Linux configuration settings.

By default, candidate mode operates in shared mode, which allows multiple users to modify the candidate configuration concurrently. When the configuration is committed in shared mode, all of the users' changes are applied.

You can optionally use candidate mode in exclusive mode, which locks out all other users from making changes to the candidate configuration.

3.3.1 Configuration candidates

When a user enters candidate mode, the system creates two copies of the running datastore: one is modifiable by the user, and the other serves as a baseline. The modifiable datastore and the baseline datastore are collectively known as a configuration candidate.

A configuration candidate can be either shared or private.

- Shared – is the default configuration candidate for CLI sessions. Multiple users can modify the shared candidate concurrently. When the configuration is committed, the changes from all of the users are applied.
- Private – is the default configuration candidate when using JSON-RPC or gNMI clients, and can optionally be used in the CLI. With a private candidate, each user modifies their own separate instance of the configuration candidate. When a user commits their changes, only the changes from that user are committed.

By default, there is a single unnamed global configuration candidate. You can optionally configure one or more named configuration candidates, which function identically to the global configuration candidate. Both shared and private configuration candidates support named versions.

3.3.2 Setting the configuration mode

After logging in to the CLI, you are initially placed in running mode. Table 1: Commands to change configuration mode describes the commands to change between modes.

Table 1: Commands to change configuration mode

To enter this mode:	Type this command:
Candidate shared	**enter candidate**
Candidate mode for named shared candidate	**enter candidate name <name>**
Candidate private	**enter candidate private**
Candidate mode for named private candidate	**enter candidate private name <name>**
Candidate exclusive	**enter candidate exclusive**
Exclusive mode for named candidate	**enter candidate exclusive name <name>**
Running	**enter running**
State	**enter state**
Show	**enter show**

Example:

For example, to change from running to candidate mode:

```
--{ running }--[  ]--
# enter candidate
--{ * candidate shared default }--[  ]--
```

The asterisk (*) next to the mode name indicates that the candidate configuration has changes that have not yet been committed.

Example

To enter candidate mode for a named configuration candidate, you specify the name of the configuration candidate. For example:

```
--{ running }--[  ]--
# enter candidate name cand1
--{ candidate shared cand1}--[  ]--
```

3.4 Committing a configuration in candidate mode

About this task

Changes made during a configuration modification session do not take effect until a **commit** command is issued. Use the **commit** command in candidate mode only.

Procedure

Step 1. Enter candidate mode:

Example

```
# enter candidate
```

Step 2. Enter configuration commands.

Step 3. Enter the **commit** command:

- To apply the changes and remain in candidate mode, enter **commit stay**.
- To apply the changes, exit candidate mode, and enter running mode, enter **commit now**.
- To apply the changes, remain in candidate mode, and save the changes to the startup configuration, enter **commit stay save**.
- To apply a comment to a **commit stay** or **save** operation, use the **comment** keyword and specify a comment.

Example

```
# enter candidate
--{ candidate shared default }--[  ]--
# interface ethernet-1/1 subinterface 1
--{ * candidate shared default }--[ interface ethernet-1/1 subinterface 1 ]--
# commit stay
All changes have been committed. Starting new transaction.
--{ + candidate shared default }--[ interface ethernet-1/1 subinterface 1 ]--
# save startup
/system:
```

```
    Saved current running configuration as initial (startup) configuration '/etc/
opt/srlinux/config.json'
--{ candidate shared default }--[ interface ethernet-1/1 subinterface 1 ]--
#
```

In this example, a user enters candidate mode and creates a subinterface for interface `ethernet-1/1`. The asterisk (*) next to `candidate shared default` in the prompt indicates that the candidate configuration has changes that have not yet been committed. After committing the changes with the **commit stay** command, the new subinterface becomes part of the running configuration.

The plus sign (+) in the prompt indicates that the currently running configuration differs from the startup configuration. The **save startup** command saves the running configuration to the startup configuration.

3.4.1 Confirming a commit operation

You can optionally configure the SR Linux to require explicit confirmation via a **tools** command for the configuration changes from a commit operation to become permanent. If the new configuration is not confirmed after a timeout period, the running datastore reverts to the previous version.

Example:

After entering configuration commands in candidate mode, use the following command to commit the configuration and start the confirmation timer:

```
--{ candidate shared default }--[  ]--
# commit confirmed
```

The **commit confirmed** command applies the changes to the running datastore and activates them. If the configuration is committed successfully, the confirmation timer is started (default 10 minutes). If you do not confirm the commit operation before the timer expires, the configuration reverts to the previous version.

Example

To confirm the commit operation and make the configuration changes permanent, enter the following command before the confirmation timer expires:

```
--{ candidate shared default }--[  ]--
# tools system configuration confirmed-accept
```

Example

To revert to the previous configuration without waiting for the confirmation timer to expire, enter the following command:

```
--{ candidate shared default }--[  ]--
# tools system configuration confirmed-reject
```

3.4.2 Validating a commit operation

You can optionally validate the configuration changes made during a commit operation before they are applied to the running datastore. The SR Linux management server checks the syntax of the changes in the candidate configuration and displays messages if validation errors are found.

Example:

Use the following command to perform a validation check for configuration changes. Note that this command only validates the changes; it does not apply them to the running datastore.

```
--{ candidate shared default }--[  ]--
# commit validate
```

If syntax errors are found in the configuration changes, SR Linux displays error messages; if validation is successful, no output is displayed.

3.4.3 Updating the baseline datastore

A configuration candidate consists of a modifiable candidate datastore and a baseline datastore, both of which are snapshots of the then-current running datastore when a user enters candidate mode.

During the lifetime of the configuration candidate, the running datastore can be modified via commits initiated from other configuration sessions. At that point, the baseline in the configuration candidate is out-of-date.

Updating the baseline datastore takes a new snapshot of the running configuration, applies the changes in the candidate datastore, and checks for configuration conflicts in the updated baseline datastore. If conflicts are found, the user is informed with a warning or error for each conflict.

A baseline datastore update is done automatically when the user commits the changes to the configuration, or can be done manually with the **baseline update** command.

Example:

In the following example, the baseline datastore in a configuration candidate is out of date, as indicated by the ! in the prompt. This indicates that another user has committed changes to the running datastore.

Entering the **baseline update** command copies the current running datastore to the baseline datastore, applies the changes in the candidate datastore, then displays any conflicts in the updated baseline datastore. If there are no conflicts, no output is returned by the command.

```
--{!* candidate shared default }--[  ]--
# baseline update
--{* candidate shared default }--[  ]--
#
```

3.5 Deleting a configuration

Use the **delete** command to delete configurations while in candidate mode.

Example:

The following example displays the system banner configuration, deletes the configured banner, then displays the resulting system banner configuration:

```
--{ candidate shared default }--[  ]--
# info system banner
    system {
        banner {
            login-banner "Welcome to SRLinux!"
        }
```

```
    }
--{ candidate shared default }--[  ]--
# delete system banner
--{ candidate shared default }--[  ]--
# info system banner
    system {
        banner {
        }
    }
```

3.6 Annotating the configuration

To aid in reading a configuration, you can add comments or descriptive annotations. The annotations are indicated by ! ! ! in displayed output.

You can enter a comment either directly from the command line or by navigating to a CLI context and entering the comment in annotate mode.

Example:

The following example adds a comment to an ACL configuration. If there is already a comment in the configuration, the new comment is appended to the existing comment.

```
--{ candidate shared default }--[  ]--
# acl ipv4-filter ip_tcp !! "Filter TCP traffic"
```

Example

To replace the existing comment, use **! ! !** instead of **! !** in the command.

The following example adds the same comment to the ACL by navigating to the context for the ACL and entering the comment in annotate mode:

```
--{ * candidate shared default }--[  ]--
# acl ipv4-filter ip_tcp
--{ * candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# annotate
Press [Meta+enter] or [Esc] followed by [Enter] to finish
-> Filter TCP traffic
```

You can enter multiple lines in annotate mode. To exit annotate mode, press Esc, then the Enter key.

Example

In CLI output, the comment is displayed in the context it was entered. For example:

```
--{ running }--[  ]--
# info acl
    acl {
        ipv4-filter ip_tcp {
            !!! Filter TCP traffic
            entry 100 {
                action {
                    drop {
                        log true
                    }
                }
            }
            entry 110 {
                action {
```

```
                    accept {
                        log true
                    }
                }
            }
        }
    }
```

To remove a comment, enter annotate mode for the context and press Esc then Enter without entering any text.

3.7 Discarding a configuration in candidate mode

You can discard previously applied configurations with the **discard** command. Use the **discard** command in candidate mode only.

- To discard the changes and remain in candidate mode with a new candidate session, enter **discard stay**.
- To discard the changes, exit candidate mode, and enter running mode, enter **discard now**.

Example:

```
All changes have been committed. Starting new transaction.

--{ candidate shared default }--[ interface ethernet-1/1 subinterface 1 ]--
# discard stay
--{ candidate shared default }--[ interface ethernet-1/1 subinterface 1 ]--
```

3.8 Displaying configuration details

The **info** command displays configuration and state information. Entering the **info** command from the root context displays the entire configuration, or the configuration for a specified context. Entering the command from within a context limits the display to the configuration under that context. Use this command in candidate or running mode.

Example:

To display the entire configuration, enter **info** from the root context:

```
--{ candidate shared default }--[  ]--
# info
<all the configuration is displayed>
--{ candidate }--[  ]--
```

Example

To display the configuration for a specific context, enter **info** and specify the context:

```
--{ candidate shared default }--[  ]--
# info system lldp
    system {
        lldp {
            admin-state enable
            hello-timer 600
            management-address mgmt0.0 {
```

```
                type [
                    IPv4
                ]
            }
            interface mgmt0 {
                admin-state disable
            }
        }
    }
--{ candidate }--[  ]--
```

Example

From a context, use the **info** command to display the configuration under that context:

```
--{ candidate shared default }--[  ]--
# system lldp
--{ candidate }--[ system lldp ]--
# info
    admin-state enable
    hello-timer 600
    management-address mgmt0.0 {
        type [
            IPv4
        ]
    }
    interface mgmt0 {
        admin-state disable
    }
--{ candidate }--[ system lldp ]--
```

Example

Use the **as json** option to display JSON-formatted output:

```
--{ candidate }--[ system lldp ]--
# info | as json
{
  "admin-state": "enable",
  "hello-timer": "600",
  "management-address": [
    {
      "subinterface": "mgmt0.0",
      "type": [
        "IPv4"
      ]
    }
  ],
  "interface": [
    {
      "name": "mgmt0",
      "admin-state": "disable"
    }
  ]
}
```

Example

Use the **detail** option to display values for all parameters, including those not specifically configured:

```
--{ candidate }--[ system lldp ]--
# info detail
    admin-state enable
```

```
    hello-timer 600
    hold-multiplier 4
    management-address mgmt0.0 {
        type [
            IPv4
        ]
    }
    interface mgmt0 {
        admin-state disable
    }
```

Example

Use the **flat** option to display the output as a series of **set** statements, omitting indentation for any sub-contexts:

```
--{ candidate }--[ system lldp ]--
# info flat
set / system lldp admin-state enable
set / system lldp hello-timer 600
set / system lldp management-address mgmt0.0
set / system lldp management-address mgmt0.0 type [ IPv4 ]
set / system lldp interface mgmt0
set / system lldp interface mgmt0 admin-state disable
```

Example

Use the **depth** option to display parameters with a specified number of sub-context levels:

```
--{ candidate }--[ system lldp ]--
# info depth 0
    admin-state enable
    hello-timer 600
```

```
--{ candidate }--[ system lldp ]--
# info depth 1
    admin-state enable
    hello-timer 600
    management-address mgmt0.0 {
        type [
            IPv4
        ]
    }
    interface mgmt0 {
        admin-state disable
    }
```

3.9 Displaying the configuration state

To display information from the state datastore, enter the **info from state** command in candidate or running mode, or the **info** command in state mode.

Example:

To display state information for a specified context from candidate or running mode:

```
--{ candidate shared default }--[  ]--
# info from state routing-policy policy bgp-export-policy
    routing-policy {
```

```
        policy bgp-export-policy {
            statement 999 {
                action {
                    accept {
                    }
                }
            }
        }
    }
```

Example

To display state information for a specified context from state mode:

```
--{ candidate shared default }--[  ]--
# enter state
--{ state }--[  ]--
# info routing-policy policy bgp-export-policy
    routing-policy {
        policy bgp-export-policy {
            statement 999 {
                action {
                    accept {
                    }
                }
            }
        }
    }
--{ state }--[  ]--
```

Example

You can change to a different mode (for example, from state mode to candidate mode), and remain in the previous context. For example:

```
--{ candidate shared default }--[  ]--
# enter state
--{ state }--[  ]--
# routing-policy policy bgp-export-policy
--{ state }--[ routing-policy policy bgp-export-policy ]--
# info
    statement 999 {
        action {
            accept {
            }
        }
    }
--{ state }--[ routing-policy policy bgp-export-policy ]--
# enter candidate
--{ candidate shared default }--[ routing-policy policy bgp-export-policy ]--
```

3.10 Saving a configuration to a file

Save the existing configuration to a file using the **save** command. Use this command in candidate or running mode.

Example:

To save the running configuration to a file:

```
--{ running }--[  ]--
# save file running-config.txt text from running
/ running configuration has been stored in 'running-config.txt'
--{ running }--[  ]--
```

Example

To save the running configuration in JSON format:

```
--{ running }--[  ]--
# save file running-config.json from running
/ running configuration has been stored in 'running-config.json'
--{ running }--[  ]--
```

Example

To save the running configuration in JSON format:

```
--{ running }--[  ]--
# save file running-config.json from running
/ running configuration has been stored in 'running-config.json'
--{ running }--[  ]--
```

Example

To save the running configuration to the initial (startup) configuration:

```
--{ + running }--[  ]--
# save startup
/ running configuration has been stored in '/etc/opt/srlinux/config.json'
--{ running }--[  ]--
```

The plus sign (+) in the prompt indicates that the running configuration differs from the startup configuration. After you enter the **save startup** command, the running configuration is synchronized with the startup configuration, and the plus sign is removed from the prompt.

3.11 Loading a configuration

Use the **load** command to load a configuration. The configuration can be from a checkpoint (see Configuration checkpoints), a JSON-formatted configuration file, the startup configuration, the factory default configuration, a rescue configuration (see Rescue configuration), or from manually entered or pasted JSON-formatted input.

Example:

To load a configuration from a checkpoint:

```
--{ * candidate shared default }--[  ]--
# load checkpoint id 0
/system/configuration/checkpoint[id=0]:
    Loaded checkpoint '/etc/opt/srlinux/checkpoint/checkpoint-0.json'
```

Example

To load a configuration from a file:

```
--{ * candidate shared default }--[  ]--
# load file home/config.txt
Loading configuration from 'home/config.txt'
```

Example

To load a rescue JSON configuration from a checkpoint:

```
--{ * candidate shared default }--[  ]--
# load rescue auto-commit
/system/configuration/checkpoint[id=__rescue__]:
    Reverting to rescue configuration
```

Example

To load a configuration from manually entered JSON-formatted input:

```
--{ * candidate shared default }--[  ]--
# system banner
--{ * candidate shared default }--[ system banner ]--
# load json
Press [Meta+enter] or [Esc] followed by [Enter] to finish
<< {
<<   "login-banner": "Welcome to SRLinux!"
<< }
```

You can enter or paste multiple lines at the << prompt in JSON-input mode. To exit JSON-input mode, press Esc, then the Enter key.

3.12 Executing configuration statements from a file

You can execute configuration statements from a source file consisting of **set** statements such as those generated by the **info flat** command (see Displaying configuration details). The SR Linux reads the file and executes each configuration statement line-by-line. You can optionally commit the configuration automatically after the file is read.

Example:

The following example executes a configuration from a specified file:

```
--{ running }--[  ]--
# source config.cfg
Sourcing commands from 'config.cfg'
Executed 20 lines in 1.6541 seconds from file config.cfg
```

Use the **auto-commit** option to commit the configuration after the commands in the source file are executed.

3.13 Configuration checkpoints

You can roll back the configuration to a previous state, known as a checkpoint. You can load a saved checkpoint into the candidate configuration, and revert the running configuration to a previously saved checkpoint.

A checkpoint is saved as a JSON-formatted file containing the complete configuration for the system. If a checkpoint file is larger than 1 Mb, it is compressed and saved in GZIP format. Checkpoint files are saved in the `/etc/opt/srlinux/checkpoint` directory. They are named `checkpoint-<number>.json` (for example, `checkpoint-0.json`), or `checkpoint-<number>.json.gz`, with the lowest number being the most recently saved checkpoint.

By default, the 10 most recent checkpoints are saved; you can configure the maximum number of checkpoint files that are kept by the system.

3.13.1 Generating a checkpoint

You can generate a checkpoint with a **tools** command or with the **save checkpoint** command. You can optionally configure the system to generate a checkpoint automatically when a configuration is committed.

Example:

The following example generates a checkpoint from the current configuration:

```
--{ !* candidate shared default }--[  ]--
# tools system configuration generate-checkpoint
/system:
    Generated checkpoint '/etc/opt/srlinux/checkpoint/checkpoint-
0.json' with name 'checkpoint-2020-10-20T23:14:12.703Z' and comment ''
```

Example

You can optionally configure a name or comment for a checkpoint; for example:

```
--{ !* candidate shared default }--[  ]--
# save checkpoint comment My-checkpoint
/system:
    Generated checkpoint '/etc/opt/srlinux/checkpoint/checkpoint-
0.json' with name 'checkpoint-2020-10-20T23:23:25.891Z' and comment 'My-checkpoint'
```

Example

The following example configures the system to generate a checkpoint automatically whenever a configuration is committed:

```
--{ !* candidate shared default }--[  ]--
# info system configuration
    system {
        configuration {
            auto-checkpoint true
            }
        }
    }
```

For automatically generated checkpoints, the comment is set to `automatic checkpoint after commit <n>`, where <*n*> is the ID of the commit that triggered the checkpoint.

3.13.2 Loading a checkpoint

Loading a checkpoint requires an interactive configuration session with a candidate session already active.

Example:

The following example loads a checkpoint into the candidate configuration. Note that you must be in candidate mode to load a checkpoint.

```
--{ * running }--[  ]--
# enter candidate
--{ * candidate shared default }--[  ]--
# tools system configuration checkpoint 0 load
/system/configuration/checkpoint[id=0]:
    Loaded checkpoint '/etc/opt/srlinux/checkpoint/checkpoint-0.json'
```

3.13.3 Reverting to a previous checkpoint

You can revert the running configuration to a previous checkpoint. When used within a candidate session, the revert operation loads the checkpoint, removing any present changes, then commits them and establishes a new candidate session.

Example:

The following example reverts the running configuration to a previously saved checkpoint:

```
--{ * running }--[  ]--
# enter candidate
--{ * candidate shared default }--[  ]--
# tools system configuration checkpoint 0 revert
/system/configuration/checkpoint[id=0]:
    Reverting to checkpoint 0 (name 'checkpoint-2019-10-14T18:47:30.282Z'
comment 'Daily_checkpoint')
/:
    Successfully reverted configuration
```

3.13.4 Clearing a checkpoint

Checkpoints can be cleared from the system either manually, with a **tools** command, or automatically when the number of saved checkpoints exceeds the configured maximum.

When the number of saved checkpoints exceeds the configured maximum, the oldest checkpoint is removed, and the number of each remaining checkpoint is incremented by 1. If you clear a checkpoint manually, the other checkpoints are not renumbered.

Example:

The following example clears a previously saved checkpoint.

```
# tools system configuration checkpoint 2 clear
/system/configuration/checkpoint[id=2]:
    Cleared checkpoint '/etc/opt/srlinux/checkpoint/checkpoint-2.json'
```

3.13.5 Configuring maximum number of checkpoints

By default, the 10 most recent checkpoints are saved; you can configure the maximum number of checkpoint files that are kept by the system.

Example:

The following example configures the system to keep a maximum of 15 checkpoint files.

```
--{ * candidate shared default }--[  ]--
# info system configuration
 system {
     configuration {
         max-checkpoints 15
         }
     }
 }
```

In this example, if 15 checkpoint files are being kept, adding a subsequent checkpoint file causes the oldest checkpoint file to be deleted and the index for the remaining checkpoint files to be incremented by 1.

3.13.6 Displaying checkpoint information

Use the **info from state** command to display information about existing checkpoints.

Example:

```
# info from state system configuration checkpoint 0
    system {
        configuration {
            checkpoint 0 {
                name checkpoint-2020-10-20T23:23:25.891Z
                comment cp002
                created 2020-10-20T23:23:25.894Z
                version v20.6.0
                username srlinux
                size 28494
            }
        }
    }
```

3.14 Rescue configuration

You can save a secondary rescue configuration to load in place of the default JSON configuration file. The rescue configuration is a checkpoint file that is loaded automatically by the management server if the default `config.json` fails when the system starts.

If both the default configuration and rescue configuration files are missing or fail, a `config.json` is regenerated and committed from the factory `config.json` that is compiled in the management server.

3.14.1 Saving a rescue configuration

You can save a rescue configuration to be loaded automatically if the default `config.json` fails when the system starts. Save the rescue configuration to a file with a **tools** command or using the **save rescue**

command. Use these commands in running mode. The system generates a `rescue-config.json` file and saves it to the `/etc/opt/srlinux/checkpoint` directory.

Example:

The following **tools** command saves a rescue configuration:

```
--{ running }--[ ]--
# tools system configuration rescue-save
/system:
    Saved current running configuration as rescue configuration '/etc/opt/srlinux/
checkpoint/rescue-config.json'
```

Example

The following example also saves a rescue configuration:

```
--{ running }--[ ]--
# save rescue
/system:
  Saved current running configuration as rescue configuration '/etc/opt/srlinux/
checkpoint/rescue-config.json'
```

Example

You can confirm the rescue configuration is saved by viewing the checkpoint directory. The following example lists the checkpoint directory:

```
--{ running }--[ ]--
# file ls /etc/opt/srlinux/checkpoint
checkpoint-0.json
rescue-config.json
```

3.14.2 Clearing a rescue configuration

To remove an existing rescue configuration, use the **rescue-clear** command to clear the configuration from the `/etc/opt/srlinux/checkpoint` directory. Use this command in running mode. You can then save a new rescue configuration to replace the cleared configuration.

Example:

The following **tools** command clears a previously saved rescue configuration:

```
--{ running }--[ ]--
# tools system configuration rescue-clear
/system:
    Cleared rescue configuration '/etc/opt/srlinux/checkpoint/rescue-config.json'
```

Example

You can confirm the rescue configuration is cleared by viewing the checkpoint directory. The following example lists the checkpoint directory:

```
--{ running }--[ ]--
# file ls /etc/opt/srlinux/checkpoint
checkpoint-0.json
```

3.15 Configuration upgrades

When the SR Linux is started following a software image upgrade, it reads the configuration in the startup `config.json` file, makes any necessary changes to ensure compatibility with the new software image, and places the upgraded configuration into the running configuration. This upgraded configuration is not saved automatically; to save the contents of the running configuration, use the following commands:

- **save startup** – Saves the running configuration to the startup configuration file, located at `/etc/opt/srlinux/config.json` (or `config.gz`).
- **save rescue** – Saves the running configuration to the rescue configuration file, located at `/etc/opt/srlinux/checkpoint/rescue-config.json` (or `rescue-config.gz`).
- **save checkpoint** – Saves the running configuration to a configuration checkpoint; for example, `/etc/opt/srlinux/checkpoint/checkpoint-0.json` (or `checkpoint-0.gz`).
- **save file <name> from running** – Saves the running configuration to the specified file in JSON format.

3.15.1 Upgrading configuration files

In addition to saving the upgraded running configuration to startup, rescue, and checkpoint configurations, you can use the following **tools** commands to upgrade existing configuration files so they are compatible with the current software version.

Example:

To upgrade the startup configuration file:

```
--{ running }--[ ]--
# tools system configuration upgrade startup
```

Example

To upgrade the rescue configuration file:

```
--{ running }--[ ]--
# tools system configuration upgrade rescue
```

Example

To upgrade a configuration checkpoint file:

```
--{ running }--[ ]--
# tools system configuration upgrade checkpoint 0
```

Example

To upgrade a specific JSON-formatted configuration file:

```
--{ running }--[ ]--
# tools system configuration upgrade file /etc/opt/srlinux/configs/myconfig.json
```

4 Securing access

The SR Linux is able to secure access to the device for users connecting via SSH or the console port, as well as for applications and FTP access. Authentication can be performed for users configured within the underlying Linux OS, and for administrative users configured within the SR Linux itself.

Depending on the user type, users are authenticated locally on the device or through interaction with the SR Linux aaa_mgr application and an authentication server group (for example, TACACS+).

4.1 User types

The SR Linux supports three user types: Linux users, local users, and remote users. Each user type is authenticated differently, as described in the following sections.

4.1.1 Linux users

Linux users are those configured in the underlying Linux OS, not in the SR Linux CLI. Information about Linux users is stored in `/etc/passwd` in the underlying Linux OS.

By default, the SR Linux has a single Linux user (username `linuxadmin`, password `NokiaSrl1!`) who has access to `sudo` to root and can run the SR Linux CLI with admin permissions. Other Linux users can be added with the `useradd` command in the underlying Linux OS.

Linux users are authenticated via the underlying Linux OS, not through the SR Linux aaa_mgr application. This means that Linux users are not subject to authentication settings configured within the SR Linux CLI, such as authentication by a TACACS+ server group.

4.1.2 Local users

Local users are users configured within the SR Linux itself. By default, SR Linux supports a single local user, named `admin`; other local users can be added as necessary.

Local users are authenticated via a gRPC interface to the aaa_mgr application. See Authentication for local users for an example configuration.

4.1.3 Remote users

Remote users are users that are not configured either in `/etc/passwd` or within the SR Linux configuration. Remote users are configured on a remote server, which is queried when the user attempts to log in to the SR Linux device.

4.2 AAA functions

The SR Linux performs authentication, authorization, and accounting (AAA) functions for each user type, as described in the following sections.

4.2.1 Authentication

For Linux users, the SR Linux authenticates via the authentication mechanism built into the underlying Linux OS.

For local users, including the SR Linux `admin` user, the SR Linux uses its gRPC interface to the aaa_mgr application for authentication. Authentication settings that apply to the local users, including a local password and TACACS+ server group, can be configured using the SR Linux CLI. See Authentication for local users for information about configuring local users.

For remote users, authentication is performed using the aaa_mgr application in coordination with the remote system.

4.2.2 Authorization

The SR Linux implements authorization through role-based access control, where each authenticated user is assigned one or more predefined roles that specify the functions the user is allowed to perform on the system. If no role is configured for a user, then the user is assigned a role that allows access to CLI plugins, but no other functions.

Authorization via role based access control is performed for all user types on all interfaces (CLI, gNMI, JSON-RPC), with the exception of the `admin` and `linuxadmin` users, which are permitted write access to all commands in the command tree. You can configure service authorization, which can limit the interface types for which the functions in a role are authorized.

You can configure the SR Linux to use information from a TACACS+ server to assign roles to an authenticated user. In this case, the priv-lvl value in the authorization-reply message received from the TACACS+ server maps to a role configured on the SR Linux. The user is assigned the role that corresponds to the priv-lvl value.

See Authorization using role-based access control for information about configuring roles for users.

4.2.3 Accounting

The SR Linux supports command accounting. Accounting records generated by the SR Linux include the entire CLI string that a user enters on the command line, including any pipes or output redirects specified in the command.

You can configure the SR Linux device to send accounting records to a destination specified in an accounting-method list, such as a TACACS+ server group or the local system.

For each user type, the SR Linux device generates accounting records as follows:

- For local users, including the SR Linux `admin` user, command accounting records are sent to the destination specified in the accounting-method list, both for commands entered in the SR Linux CLI and for commands entered in the bash shell.

- For Linux users and remote users, command accounting records are sent for commands entered in the SR Linux CLI (including Linux commands entered in the SR Linux CLI using the **bash** command), although not for commands entered in the bash shell.

See Configuring accounting for an example configuration.

4.3 AAA server group configuration

The SR Linux supports the following server group types for AAA functions:

- local – Uses local authentication, including `/etc/passwd`, `/etc/group`, and logging via syslog.
- TACACS+ – Performs authentication, authorization, and accounting via interaction with servers in a TACACS+ server group.

Users whose AAA functions are handled by the aaa_mgr application (that is, the SR Linux `admin` user and local users) can use one of these server groups for authentication and accounting.

The TACACS+ server group can have up to five servers. When authenticating a user or writing an accounting record, the SR Linux tries each server in the group in a round-robin fashion until a response is received. If no response is received within a specified timeout period, the SR Linux tries the next server in the group.

If no response is received from any of the servers in the group, the SR Linux moves to the next specified authentication or accounting method. If no other method is specified, or the TACACS+ server group is the last method in the list, then the authentication or accounting request is rejected.

4.3.1 Configuring an AAA server group

The following example shows settings for a TACACS+ and a local server group to be used for AAA functions. TACACS+ requests are sourced from the mgmt network-instance.

The TACACS+ server group consists of three TACACS+ servers. The timeout period specifies that the SR Linux wait 30 seconds for a response from a server before trying the next server in the group.

For the server group of type local, no external servers can be specified. The local server group uses `/etc/passwd` and `/etc/group` for authentication, and syslog for accounting. The timeout period specifies that the SR Linux wait a maximum of 60 seconds for an AAA function to complete.

Example:

```
--{ * candidate shared default }--[  ]--
# info system aaa
 system {
     aaa {
         server-group tacacs-all {
            type tacacs
            timeout 30
            server 1.2.2.1 {
                network-instance mgmt
                tacacs {
                    secret-key $aes$3/Iz5veTDRV0=$6GxkrGjFbqWbYMA0T3itgQ==
                }
            }
            server 1.2.2.2 {
                network-instance mgmt
                tacacs {
                    secret-key $aes$3/Iz5veTDRV0=$6GxkrGjFbqWbYMA0T3itgQ==
```

```
                }
            }
            server 1.2.2.3 {
                network-instance mgmt
                tacacs {
                    secret-key $aes$3/Iz5veTDRV0=$6GxkrGjFbqWbYMA0T3itgQ==
                }
            }
         server-group local {
            type local
            timeout 60
            }
         }
     }
 }
```

4.4 Authentication for local users

Local users are those configured within the SR Linux CLI. For a local user, you can configure a password and specify one or more authentication methods, including local authentication or a TACACS+ server group.

4.4.1 Configuring authentication for local users

By default, there is a single local user configured on the SR Linux, `admin`. You can configure additional local users.

Example:

The following example configures a password for the SR Linux `admin` user:

```
--{ * candidate shared default }--[  ]--
# system aaa authentication admin-user password NewPass1234
```

Example

The following example creates a local user called `sruser` and configures a password:

```
--{ * candidate shared default }--[  ]--
# system aaa authentication user srlinux password sr1l234
```

When the user configuration is displayed, the password is hashed; for example:

```
--{ * candidate shared default }--[  ]--
# info system aaa
    system {
        aaa {
            authentication {
                user srlinux {
                    password $ar2$KGSITqfJu5g=$iZZ6XKXtXOZOGMbeXO2ypg==
                }
            }
        }
    }
```

Example

To change an existing user's password, use the same command that created the user and configure the new password; for example:

```
--{ * candidate shared default }--[  ]--
# system aaa authentication user srlinux password NewPassword1234
```

Example

The following example specifies authentication methods. When a user attempts to log in, the user is authenticated using local authentication first. If local authentication fails, the SR Linux tries the servers in TACACS+ server group tacacs-all. If the user cannot be authenticated through either method, the authentication attempt is rejected.

```
--{ * candidate shared default }--[  ]--
# info system aaa
 system {
     aaa {
         authentication {
            authentication-method [
                local
                tacacs-all
            ]
         }
     }
 }
```

4.5 Authorization using role-based access control

Authorization is performed through role-based access control. Users can be configured with a set of one or more roles that indicate the privileges for which they are authorized in the system.

A role consists of one or more rules, which specify a schema path the role can have privileges for, and a corresponding action, which can be read, write, or deny. After authentication, a user is authorized to perform the specified action defined in the path for the role the user is assigned.

Role-based access control supports service authorization, which allows you to limit the actions a user is authorized to perform to specific access types such as CLI and gNMI.

4.5.1 Role configuration

Roles consist of a set of rules that define a schema path-reference and a corresponding action. The path-reference specifies system functions that are subject to authorization, and the action specifies the privilege type for users assigned the role: read, write, or deny.

The path-reference is specified relative to the root level. For example, the path-reference "`/`" indicates all CLI functions at the root level and below; the path-reference "`/system`" indicates all CLI functions at the `system` level and below, and the path-reference "/system configuration" indicates all CLI functions at the `system configuration` level and below.

If no role is assigned to a user at login, the user has access to all CLI plugins, but no access to CLI commands at any level. This is equivalent to a rule with `/` as the path-reference and `deny` as the action.

Up to 32 roles are supported; each role can have a list of up to 256 paths. The order of the path-references in a role does not matter; the longest match is used when validating a command against a path-reference.

The syntax for the path-reference is SR Linux CLI format, with "/" representing the root. Wildcards and ranges can be used with path-references, in the same way they can in CLI syntax.

Each entry within the path-reference should use double quotes, unless the command string is a single word with no spaces. If the path itself includes quotes, use backslash characters to indicate the quotes. For example, to specify the following in a path-reference:

`a "b"` and `c "d"`

You can configure the following for the path-reference:

path-reference `"a \"b\"" "c \"d\""`

4.5.1.1 Configuring a role

To configure a role, you define one or more rules that specify a path indicating the command string that is subject to authorization, and a corresponding action such as read, write, or deny.

Example:

The following is an example of creating a role named `testrole` that contains two rules: one rule to limit access to network-instance `red`, and another rule to give write access to the rest of the tree. A user assigned this role upon authentication is able to configure everything in the system except network-instance `red`.

First, define the role under the `system aaa authorization` context:

```
--{ * candidate shared default }--[  ]--
# info system aaa
    system {
        aaa {
            authorization {
                role testrole {
                }
            }
        }
    }
```

Then specify the rules under the `system configuration role` context:

```
--{ * candidate shared default }--[  ]--
# info system configuration
    system {
        configuration {
            role testrole {
                rule / {
                    action write
                }
                rule "network-instance red" {
                    action read
                }
            }
        }
    }
```

Example

The following example configures a role named `acl_app` that allows a user to view the state of the acl_mgr application, but no other information:

```
--{ * candidate shared default }--[  ]--
# info system aaa
    system {
        aaa {
            authorization {
                role acl_app {
                }
            }
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# info system configuration
    system {
        configuration {
            role acl_app {
                rule system {
                    action deny
                }
                rule "system app-management application acl_mgr state" {
                    action read
                }
            }
        }
    }
```

When a user that is assigned the `acl_app` role attempts to read information from the system state datastore, only the state of the acl_mgr application is displayed. For example:

```
--{ * candidate shared default }--[  ]--
# info from state system application acl_mgr
system {
    app-management {
        application acl_mgr {
            state running
        }
    }
}
```

4.5.2 Assigning roles to users

A user can be assigned one or more roles. The rules configured in the user's roles specify the commands the user is authorized to issue.

Up to 32 roles can be assigned to a user. If a user has multiple roles assigned, all of the rules configured in all of the roles apply to the user. If multiple roles reference the same path, the most specific rule is used.

For service authorization, the system merges all roles that have the service (CLI, gNMI, and so on) the user has logged in with, and excludes any roles that omit the service.

Example:

In the following example, the user `srlinux` is assigned the role `testrole`.

After the `srlinux` user is authenticated, the user is authorized to use the system according to the rules configured in the role `testrole`. Using the example from Configuring a role, this would authorize the `srlinux` user to configure everything in the system except for the network-instance `red`.

```
--{ * candidate shared default }--[  ]--
# info system aaa
    system {
        aaa {
            authentication {
                user srlinux {
                    role [
                        testrole
                    ]
                }
            }
        }
    }
```

4.5.3 Authorization using a TACACS+ server

You can configure the SR Linux to use a TACACS+ server to provide authorization for role-based access control. When TACACS+ authorization is configured, the actions an authenticated user can perform depend on the priv-lvl value configured for the user on a TACACS+ server. If **priv-lvl-authorization** is not set to **true**, the authenticated user is authorized as an admin user.

TACACS+ authorization for SR Linux users works as follows:

1. After a user is authenticated, the SR Linux sends the TACACS+ server an authorization-request message based on a shell (exec) session starting. This authorization-request is sent immediately following user authentication, but before the shell is started.
2. The TACACS+ server returns an authorization-reply message that includes the priv-lvl value configured for the user for shell access.
3. On the SR Linux, roles have been configured that map to a priv-lvl value.
4. The SR Linux allows the user to perform actions specified in roles with a priv-lvl value equal or lower to the priv-lvl value returned by the TACACS+ server.

4.5.3.1 Configuring TACACS+ Authorization

To configure TACACS+ authorization for SR Linux users, you enable priv-lvl authorization for the TACACS + server group and configure roles that specify the priv-lvl mapping for each role subject to authorization using a TACACS+ server.

Example:

The following configuration enables priv-lvl authorization for the `tacacs-all` server group:

```
--{ * candidate shared default }--[  ]--
# info system aaa server-group tacacs-all
    system {
        aaa {
            server-group tacacs-all {
                priv-lvl-authorization true
            }
        }
```

```
    }
```

Example

The following configuration specifies two roles, `interface oper-state` and `network-instance oper-state`. The `interface oper-state` role grants read access for all interface oper-states, and the `network-instance oper-state` role grants read access for all network-instance oper-states.

```
--{ * candidate shared default }--[  ]--
# info system configuration role *
    system {
        configuration {
            role "interface oper-state" {
                rule "interface * oper-state" {
                    action read
                }
            }
            role "network-instance oper-state" {
                rule "network-instance * oper-state" {
                    action read
                }
            }
        }
```

Example

The following configuration specifies the priv-lvl mapping for both roles.

In this example, when a user is authenticated, the SR Linux sends an authorization-request based on a shell (exec) session starting. The TACACS+ server returns an authorization-reply that specifies the priv-lvl value for the shell.

If the priv-lvl value returned by the TACACS+ server is 14, the user is assigned both roles; that is, read access to all interface oper-states and all network-instance oper-states, but no access to anything else in the system.

If the priv-lvl value returned by the TACACS+ server is 13, the user is assigned only the `network-instance oper-state` role and can read the oper-state for network-instances, but has no access to anything else in the system.

```
--{ * candidate shared default }--[  ]--
# info system aaa authorization
    system {
        aaa {
            authorization {
                role "interface oper-state" {
                    tacacs {
                        priv-lvl 14
                    }
                }
                role "network-instance oper-state" {
                    tacacs {
                        priv-lvl 13
                    }
                }
            }
        }
    }
```

Example

The following example configures service authorization for a role that uses TACACS+ authorization. See Configuring service authorization.

In this example, when a user is authenticated by a TACACS+ server in the server group `tacacs-all`, if the TACACS server returns priv-lvl 15 for the user, then `role_1` is assigned to the user.

Service authorization is configured for the role so that all services except gNMI are authorized for users assigned this role. This means users assigned `role_1` are authorized for the functionality defined in the rules of `role_1` if they connect using the CLI, JSON-RPC, or FTP, but are not authenticated if they connect using gNMI.

```
--{ * candidate shared default }--[  ]--
# info system aaa authorization role role_1
    system {
        aaa {
            authorization {
                role role_1 {
                    services [
                        cli
                        json-rpc
                        ftp
                    ]
                    tacacs {
                        priv-lvl 15
                    }
                }
            }
        }
    }
```

When a user is assigned multiple roles, the user is authorized for all services specified in the roles they are assigned, according to the rules defined in the roles. For example, if a user is assigned `role_1`, which allows access to the system via CLI, and also assigned `role_2`, which allows access via gNMI, the user is authorized to use the CLI to perform the functions defined in the rules of `role_1`, and is authorized to use the gNMI interface for the functions defined in the rules of `role_2`.

4.5.4 Service authorization for local users

Service authorization allows you to block or allow access to a user depending on the interface they use to connect to the device. You can use service authorization to restrict access to a controller, allowing it to speak through programmatic interfaces, but without credentials that can be used by someone logged into the CLI.

To configure service authorization, you assign roles to local users that authorize them to issue commands using specified services, such as the CLI, gNMI, or JSON-RPC interfaces. For example, you can define a role that grants read access to subinterface statistics, and limits access to that information to the gNMI service. When you assign that role to a user, the user is allowed to read subinterface information via the gNMI interface only.

The following interface types can be configured for service authorization:

- `cli` - access to the system via CLI
- `gnmi` - access to the system via gNMI
- `json-rpc` - access to the system via JSON-RPC

- `ftp` - access to the system via FTP

After a user is authenticated, the system checks whether the user is assigned a role that allows access via the interface they used to connect to the device. For example, if a user connects using gNMI, the system checks whether the user is assigned any roles that authorize access via the gNMI interface. If the user is assigned a role that authorizes access via gNMI, the user receives access to the system via gNMI according to the rules defined in the set of roles that includes gNMI. If the user connects via gNMI, but is not assigned any role that authorizes access via gNMI, the authorization attempt is rejected and the session is closed.

If a user is assigned multiple roles, the user is authorized for all services specified in the roles they are assigned, according to the rules defined in the roles. For example, consider a user assigned role r1, which allows access to the system via CLI, and also assigned role r2, which allows access via gNMI. The user is authorized to use the CLI to perform the functions defined in the rules of role r1, and is authorized to use the gNMI interface for the functions defined in the rules of role r2.

By default, a role has no services configured for authorization. When you configure a role, you must specify the services that apply to the role. Users assigned the role are authorized to perform the functions defined in the role using the specified services.

4.5.4.1 Configuring service authorization

To configure service authorization, you create a role that defines rules permitting or denying access to system functionality, assign the role to one or more users, and specify the services over which users assigned the role are authorized to perform the rules defined in the role.

Example:

The following example creates a role `read-oper-state` that allows reading subinterface oper-state but nothing else:

```
--{ * candidate shared default }--[  ]--
# info system configuration role *
    system {
        configuration {
            role read-oper-state {
                rule / {
                    action deny
                }
                rule "interface * subinterface * oper-state" {
                    action read
                }
            }
        }
    }
```

Example

The following example creates a user `gnmiuser` that is assigned the `read-oper-state` role when authenticated:

```
--{ * candidate shared default }--[  ]--
# info system aaa authentication
    system {
        aaa {
            authentication {
                authentication-method [
                    local
                ]
```

```
                user gnmiuser {
                    role [
                        read-oper-state
                    ]
                }
            }
        }
    }
```

Example

The following example configures service authorization so that users assigned the `read-oper-state` role, such as `gnmiuser`, can perform the functionality defined in the role only if they have connected to the system via gNMI. If the user connects via a different service, such as by logging into the CLI, the user is not authorized for the functionality defined in the role.

```
--{ * candidate shared default }--[  ]--
# info system aaa authorization
    system {
        aaa {
            authorization {
                role read-oper-state {
                    services [
                        gnmi
                    ]
                }
            }
        }
    }
```

4.6 Accounting configuration

When accounting is enabled, the SR Linux device generates command accounting records as described in Accounting.

The following is an example of accounting records generated by the SR Linux device:

```
Aug 7 22:34:09
127.0.0.1 bob ssh 172.17.0.1 start task_id=2 timezone=UTC service=shell priv-lvl=15
cmd=tail -f /var/log/tac_plus.acct
Aug 7 22:34:09
127.0.0.1 bob ssh 172.17.0.1 stop task_id=2 timezone=UTC service=shell priv-lvl=15
cmd=tail -f /var/log/tac_plus.acct
Aug 7 22:34:14
127.0.0.1 bob ssh 172.17.0.1 start task_id=5 timezone=UTC service=shell priv-lvl=15
cmd=help
Aug 7 22:34:14
127.0.0.1 bob ssh 172.17.0.1 stop task_id=5 timezone=UTC service=shell priv-lvl=15
cmd=help
```

4.6.1 Configuring accounting

The following example configures accounting records to be sent to the tacacs-all server group. The SR Linux generates an accounting record when a command is started and when it is stopped.

Example:

```
--{ * candidate shared default }--[  ]--
# info system aaa
 system {
     aaa {
         accounting {
            accounting-method [
                tacacs-all
            ]
            event commands {
                record start-stop
            }
         }
     }
 }
```

4.7 Displaying user session information

To display information about users currently logged in to the SR Linux device, use the **show system aaa authentication session** command.

Example:

```
# show system aaa authentication session
+----+-------+-------------+----------------+----------+-------+-------------+----------------------+
| ID | User  |   Service   | Authentication | Priv-lvl |  TTY  | Remote  |      Login time         |
|    | name  |    name     |     method     |          |       |  host   |                         |
+====+=======+=============+================+==========+=======+=========+=========================+
| 4  | bob   | srlinux-cli |    tacacs      |    15    | pts/1 | 2.1.0.2 | 2021-12-06T21:24:07.80Z |
| 11 | user* | srlinux-cli |    local       |          | pts/4 |         | 2021-12-07T04:06:06.93Z |
+----+-------+-------------+----------------+----------+-------+---------+-------------------------+
```

The asterisk (*) next to the username indicates the session from which the **show system aaa authentication session** was invoked.

4.8 Disconnecting user sessions

To disconnect a user currently logged in to the SR Linux device, use the **tools system disconnect session-id <session-id>** command and specify the session ID of the user. To list the session IDs of active users, enter the **show system aaa authentication session** command.

Example:

```
# show system aaa authentication session
+----+-------+-------------+----------------+----------+-------+-------------+----------------------+
| ID | User  |   Service   | Authentication | Priv-lvl |  TTY  | Remote  |      Login time         |
|    | name  |    name     |     method     |          |       |  host   |                         |
+====+=======+=============+================+==========+=======+=========+=========================+
| 6  | bob   | srlinux-cli |    tacacs      |    15    | pts/1 | 2.1.0.2 | 2021-12-06T21:24:07.80Z |
| 11 | user* | srlinux-cli |    local       |          | pts/4 |         | 2021-12-07T04:06:06.93Z |
+----+-------+-------------+----------------+----------+-------+---------+-------------------------+

# tools system disconnect session-id 6
Terminating cli session 6 owned by user 'bob' logged from pts/1
/system/aaa/authentication/session[id=6]:
```

```
Disconnecting aaa cli session(s): 6
```

4.9 Configuring idle-timeout for user sessions

You can configure the idle-timeout for user sessions, which disconnects a user session after a specified period of inactivity. By default, user sessions are disconnected after 15 minutes of inactivity.

The idle-timeout setting applies to SR Linux users and remote users. It does not apply to Linux users or to JSON-RPC or gNMI client sessions.

After a user session has been inactive for one-half of the idle-timeout period, a notification is displayed indicating that the user will be logged out if the session remains idle for the remainder of the idle-timeout period.

Example:

The following example configures the idle-timeout so that SR Linux user sessions and remote user sessions are disconnected after 20 minutes of inactivity:

```
--{ * candidate shared default }--[  ]--
# info system aaa
 system {
     aaa {
         authentication {
            idle-timeout 20
            }
         }
     }
 }
```

5 Management servers

You can configure the following management servers on the SR Linux:

- gNMI server – allows external gNMI clients to connect to the device and modify the configuration and collect state information.
- JSON-RPC server – allows you to issue JSON-formatted requests to the device to retrieve and set configuration and state.

You can configure Transport Layer Security (TLS) profiles, which contain TLS settings that can be provided to the gNMI and JSON-RPC management servers.

5.1 gNMI server

The SR Linux device can enable a gNMI server that allows external gNMI clients to connect to the device and modify the configuration and collect state information.

When the gNMI server is enabled, the SR Linux gnmi_mgr application functions as a target for gNMI clients. The gnmi_mgr application validates gNMI clients and passes Get, Set, and Subscribe RPCs to the SR Linux mgmt_svr application via the gRPC interface. See the *SR Linux System Management Guide* for details about the supported RPCs.

Configuration changes made by gNMI clients are made within a private candidate configuration, using a snapshot of the current running configuration as a baseline for the private candidate. As with other types of candidate configurations, the private candidate can operate in exclusive mode, which locks out other users from concurrently modifying the private candidate configuration.

Sessions between gNMI clients and the SR Linux device must be encrypted using TLS. You can specify TLS settings within a TLS profile, and apply the TLS profile when configuring a gNMI server within a network-instance. When the gNMI server is enabled, gNMI clients connect and authenticate to the SR Linux device using the settings specified in the TLS profile.

New connections between gNMI clients and the SR Linux device are mutually authenticated. By default, the SR Linux device validates the X.509 certificate of the gNMI client, and the other way around; this behavior can be disabled in the TLS profile. The SR Linux device, after validating the X.509 certificate of the gNMI client, performs local authentication, if the **use-authentication** parameter is set to **true**. In this case, the gNMI client is required to provide a username and password in the metadata of the Get, Set, and Subscribe RPCs. The supplied username and password are authenticated by the SR Linux aaa_mgr application.

See the *SR Linux System Management Guide* for examples of using the gnmi_cli, gnmi_get, and gnmi_set open source gNMI clients to configure and retrieve state information about the SR Linux device.

5.1.1 Configuring a gNMI server

The SR Linux supports configuring a gNMI server under one or more network-instances. You can specify limits for the number of simultaneous active gNMI client sessions, as well as the number of connection attempts per minute by gNMI clients.

For the network-instance where the gNMI server is running, you can specify the IP address and port for gNMI client connections, as well as the TLS profile used for authenticating gNMI clients. See TLS profiles.

Example:

The following example shows a configuration that enables a gNMI server on the SR Linux device. The gNMI server is configured so that gNMI clients can connect to the SR Linux via the mgmt network-instance on port 50052. Connecting gNMI clients are authenticated using the settings specified in the TLS profile `tls-profile-1`.

```
--{ * candidate shared default }--[  ]--
# info system gnmi-server
 system {
     gnmi-server {
         admin-state enable
         timeout 7200
         rate-limit 60
         session-limit 20
         network-instance mgmt {
             admin-state enable
             use-authentication true
             port 50052
             tls-profile tls-profile-1
             source-address [
                 ::
             ]
         }
     }
 }
```

5.2 JSON-RPC server

You can enable a JSON-RPC server on the SR Linux device, which allows you to issue JSON-formatted requests to the device to retrieve and set configuration and state. You can use the JSON-RPC API to run CLI commands and standard get and set methods. The SR Linux device returns responses in JSON-format.

Configuration changes made using the JSON-RPC API are made within a private candidate configuration, using a snapshot of the current running configuration as a baseline for the private candidate. As with other types of candidate configurations, the private candidate can operate in exclusive mode, which locks out other users from concurrently modifying the private candidate configuration.

When the JSON-RPC server is enabled, the application passes the requests to the SR Linux mgmt_svr application via the gRPC interface.This JSON-RPC API uses HTTP and HTTPS for transport, and users are authenticated with the aaa_mgr application. HTTPS requests can be authenticated using TLS, using settings specified in a TLS profile. See TLS profiles.

See the *SR Linux System Management Guide* for examples of using the get method to retrieve state information from the SR Linux, the set method to modify the SR Linux configuration, and the cli method to enter SR Linux CLI commands.

5.2.1 Configuring a JSON-RPC server

The SR Linux supports configuring a JSON-RPC server under one or more network-instances. You can specify limits for the number of simultaneous active HTTP or HTTPS connections and the TCP port used

for HTTP or HTTPS connections. If the TCP port is in use when the JSON-RPC server attempts to bind to it, the commit operation fails.

Example:

The following example shows a configuration that enables a JSON-RPC server within the mgmt network-instance on the SR Linux device. The JSON-RPC server is configured so that HTTP requests are accepted on TCP port 4000, and HTTPS requests are accepted on TCP port 443. HTTPS requests are authenticated using the settings in the TLS profile `tls-profile-1`.

```
--{ * candidate shared default }--[  ]--
# info system json-rpc-server
 system {
     json-rpc-server {
         admin-state enable {
         network-instance mgmt {
             http {
                 admin-state enable
                 use-authentication true
                 session-limit 1
                 port 4000
             }
             https {
                 admin-state enable
                 use-authentication true
                 session-limit 1
                 port 443
                 tls-profile tls-profile-1
             source-address [
                 ::
             ]
             }
     }
 }
```

5.3 TLS profiles

Transport Layer Security (TLS) is a protocol for enabling applications or devices to exchange information. The SR Linux supports configuring TLS settings in TLS profiles, which can be provided to applications such as the gNMI server and the JSON-RPC server, so that clients connecting to the SR Linux device via these applications are authenticated using the settings in the TLS profile.

5.3.1 Configuring a TLS profile

A TLS profile can specify whether authentication is performed for clients connecting to SR Linux applications to which the profile is applied. Within a TLS profile, you can configure certificates, keys, and ciphers to use when negotiating TLS connections with clients.

Example:

```
--{ * candidate shared default }--[  ]--
# info system tls
 system {
     tls {
         server-profile tls-profile-1 {
                key $aes$4NAaR4Zz5skA=$3Pv773cUer0TaPNHqRQ==
                certificate "-----BEGIN CERTIFICATE REQUEST-----
```

```
MIIEUjCCAjoCAQAwDTELMAkGA1UEBhMCVVMwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQDxG7nZ
qw7EjHM6uBit7Bi1gO/PgQTD2OqJTOtFqEXcFzGcbeeFk903v7OTNws0h11A1pZGnT9fPXv/
hYmAcvOv4FZz99jrGZ8WiwJMpxG+wG3rNdCwEfc59cqFOu9k8pbxLZVYck+pVcjgUK9wLZMHtDVYADXp7I5h
NYMLdettpSSucXaZcWpTzD/xJtePa9dkQ75LGcl/
JMivhx+7EYsbBRlkoSeluMByGavxjefNPJFtvOUAPE7CvdMprntHA7op5FkSoqZcoTzA0V1BcaNtblU7j8DL
1UnsRk6Mi9M5Sz5McQDW8cn223pT9AceLCM27LY62rNEcpAZHNumPqG352+mdLqZ7sJmfwScB3EISj6YV+ni
sZ+K8woR7PD0oz3rsnYGjjtE3xf/
NwXNdioFaVF80nkhwbpoSYZFuwzigkBvsyeUQzmYe4ehC5eFezmWracItmsqzjsDqoJZa5y3ngAK72ag9wQN
2Lz3AHYvMlC3Gn4D2P/
oNH1ywL0DeSdEMxukQamSF8EiEvCu1cBkBhab3blVOp61c7IsaNHdW1k95cFpfNLT+1HoWJlkaIVI7gCTgPW
2UURy67lUBAl4rdpFnnkRLnJfKYgjScLWSw0ur17NOTinflAgx4k5AXZP8FpvW3S0KyqR0S8UHajcsYsRd7W
aIVOhFOeXUxYeGQIDAQABoAAwDQYJKoZIhvcNAQELBQADggIBAFdyqd7yGmbM9E12i4y7nwUvORg5nwr6b5Z
aCGnCl3h8bYerhS/QjNTahAJYKl+G2pYCOdfq435B8NNFdsEfBJ9Z8C6Vs/
kixlfFVGGZVaglxR9Vgs13Srht0aE4LgE2BiUwJe5szicGRr7M/OXCN/
yS2fH5qbuHyJHdP3UiEliNleuYLO+U0ALN0XXVrFJ+FF+eyoFNKGETl+tg2Ruevh7tuVgMLD6jFhZwkm7hkS
eoG4h22rMYQ0EEJc/rjYuwT/xZySOhIIwbpg/TdDtoQi7j4Cru0b5BibZkcfRxQDhzqIvAPi7U113i7qPq/
jiC2fqnRoxl2lVuPuwCyEhDnWHatBT/
oIPEpJfvg7+zvnGk3PHvlpH3G6A85oEWFa8tbCkCt9ca8exwJCmbNCEbBnWL/
tl75HOGNfTweqNxoggQNbLEG8mmxDu9t5e/
LSDyeDycJ0CE2f8kFh+E7RGw6xznyUtS9c0CasOi0kAeXZ9fm1JRrYlAR+ADYECvRYmQEOfgpio/
2+vjTmBU0rmTSZkoNY5Ijw+SYljCzgjc9JX9Rt+EpRqnYm3BFBCg0yzW2bLyKFOZSYzbuHr0NXpJY50V04An
/Glj6Cv/vjPE8wUTkUecRBwYuyezNBPY7m7AU6sd1hTMcL3sDTJ2pzEJT56wKcV9zjnoQyBmrwatPjpU----
-END CERTIFICATE REQUEST-----"
                authenticate-client true
                cipher-list [
                    aes128-sha256
                    des-cbc3-sha
                    camellia128-sha
                ]
            }
        }
    }
```

5.3.2 Generating a self-signed certificate

From the SR Linux CLI, you can create a self-signed certificate and key. By default, SHA256 with RSA encryption are used for the signature algorithm. Private keys are not encrypted using DES/3, and are 4096 bits in length.

Example:

The following example generates a private key, followed by the self signed certificate:

```
# tools system tls generate-self-signed email info@nokia.com country us organization nokia
/system/tls:
-----BEGIN PRIVATE KEY-----
MIIJRAIBADANBgkqhkiG9w0BAQEFAASCCS4wggkqAgEAAoICAQC3Q7PnCWjevlFn
--snip--
1ecXRjvpRvIEDUWYqVOioMfCaWxJoTJUX6HgjIY0Z9ktfWiceX1Ka4e3ZxIpEC4p
SvzKsoCTqpQcIk5pCsALhznOO6ArtPc4
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIE/TCCAuWgAwIBAgIJAKzAUREbPIV1MA0GCSqGSIb3DQEBCwUAMBQxEjAQBgNV
--snip-
BTTcAiQnIIkdJ0niqE+ZcOneSgxP3dKEovWQ+Bh3ES2QLsqbDvBYjz4eBoDigaAZ
KDnK207h3hiKLAaxMbAQeWGiu2ZKoKKdd4QE6OphOw6T
-----END CERTIFICATE-----
```

This **tools** command example is equivalent to the following **openssl** command:

```
# openssl req -x509 -newkey rsa:4096 -days 365
```

5.3.3 Generating a certificate signing request

You can issue a CLI command that returns a private key and certificate signing request (CSR). This CSR can be passed to a certificate authority (the same one that the client/server uses to validate certificates on either side) for the certificate authority to sign the request; the CSR cannot be used as-is.

Example:

The following command returns the private key, followed by the CSR:

```
# tools system tls generate-csr email info@nokia.com country us organization nokia
/system/tls:
-----BEGIN PRIVATE KEY-----
MIIJRAIBADANBgkqhkiG9w0BAQEFAASCCS4wggkqAgEAAoICAQC3Q7PnCWjevlFn
--snip--
1ecXRjvpRvIEDUWYqVOioMfCaWxJoTJUX6HgjIY0Z9ktfWiceX1Ka4e3ZxIpEC4p
SvzKsoCTqpQcIk5pCsALhznOO6ArtPc4
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE REQUEST-----
MIIC5TCCAc0CAQAwgZ8xCzAJBgNVBAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlh
vy3MjE7rJtmWTg0pTfiuU4BFoAzLhHRN1hl1mXuE2m6XJ8gvBPp2sN7SvieUCy/L
RVTs+/Fmcc4vMjx3t/0hAewIsd7DNe+kVQ==
-----END CERTIFICATE REQUEST-----
```

This **tools** command example is equivalent to the following **openssl** command:

```
# openssl req -newkey rsa:4096
```

6 Logging

The SR Linux device implements logging via the standard Linux syslog libraries. The SR Linux device uses rsyslog in the underlying Linux OS to filter logs and pass them on to remote servers or other specified destinations.

The main configuration file for rsyslog is `/etc/rsyslog.conf`. The SR Linux installs a minimal version of the `/etc/rsyslog.conf` file, and maintains an SR Linux-specific configuration file in the `/etc/rsyslog.d/` directory. Per-application logging configuration files are also kept in the `/etc/rsyslog.d/` directory; these files are named `/etc/rsyslog.d/`*nn-app*`.conf`.

You can modify the SR Linux logging configuration using the CLI, northbound API, or by editing the `.conf` files manually. The SR Linux device overwrites any parts of the configuration that are owned by SR Linux, and this may supersede any configuration done manually.

The SR Linux `.conf` files in `/etc/rsyslog.d` use standard rsyslog syntax for configuring filters and actions within rules.

The SR Linux supports configuration of Linux facilities and SR Linux subsystems as sources for log messages to filter. See the *SR Linux Log Events Guide* for properties and descriptions of the log messages that can be generated by SR Linux subsystems.

Basic logging configuration consists of specifying a source for input log messages, filtering the log messages, and specifying an output destination for the filtered log messages.

6.1 Input sources for log messages

The SR Linux supports using messages logged to Linux syslog facilities and messages generated by SR Linux subsystems as input sources for log messages.

Table 2: Linux syslog facilities describes the Linux syslog facilities that can be used as input sources for log messages.

Table 2: Linux syslog facilities

Facility	Description
auth	Security/authorization messages that do not contain secret information
authpriv	Security/authorization messages that may contain secret information
cron	Messages generated by cron
daemon	Messages generated by system daemons without their own facility
ftp	Messages generated by an FTP daemon
kern	Messages generated by the kernel
local0	Local use 0

Facility	Description
local1	Local use 1
local2	Local use 2
local3	Local use 3
local4	Local use 4
local5	Local use 5
local6	Local use 6
local7	Local use 7
lpr	Messages generated by the line printer subsystem
mail	Messages generated by a mail client or server
news	Messages generated by the network news subsystem
syslog	Messages generated internally by syslog
user	Messages generated by a user
uucp	Messages generated by the UUCP (UNIX-to-UNIX copy) subsystem

Table 3: SR Linux logging subsystem names lists the SR Linux subsystems that produce messages that can serve as input sources for log messages. By default, SR Linux subsystem messages are logged to Linux syslog facility local6.

Table 3: SR Linux logging subsystem names

Subsystem	Description
aaa	Messages generated by the aaa_mgr application (not including accounting messages)
accounting	Accounting messages generated by the aaa_mgr application
acl	Messages generated through an ACL log action
app	Messages generated by the aaa_mgr application
arpnd	Messages generated by the arp_nd_mgr application
bfd	Messages generated by the bfd_mgr application
bgp	Messages generated by the bgp_mgr application
chassis	Messages generated by the chassis_mgr application

Subsystem	Description
fib	Messages generated by the fib_mgr application
gnmi	Messages generated by the gnmi_server application
json	Messages generated by the json_rpc_server application
linux	Messages generated by the linux_mgr application
lldp	Messages generated by the lldp_mgr application
mgmt	Messages generated by the mgmt_svr application
mpls	Messages generated by the mpls_mgr application
netinst	Messages generated by the net_inst_mgr application
policy	Messages generated by the policy_mgr application
sdk	Messages generated by the sdk_mgr application
staticroute	Messages generated by the static_route_mgr application
xdp	Messages generated by the xdp_mgr application

6.2 Filters for log messages

You can configure filters to target specific messages or groups of log messages captured within the input message source.

A filter can specify the set of messages generated by a Linux facility at a specified priority; for example, messages generated by the kernel that have a priority of warning or higher, or mail facility messages that have a priority of critical.

Filtering can be done for messages generated by a specific SR Linux subsystem; for example, messages generated by the aaa_mgr application, or messages generated by the chassis_mgr application. SR Linux subsystem messages go to a specified Linux facility (by default, local6), and you can create filters for subsystem-specific messages from this facility.

Table 4: Logging priorities describes the logging priorities in order of severity.

Table 4: Logging priorities

Code	Priority name	Description
0	emergency	System is unusable
1	alert	Action must be taken immediately
2	critical	Critical conditions

Code	Priority name	Description
3	error	Error conditions
4	warning	Warning conditions
5	notice	Normal but significant conditions
6	informational	Informational messages
7	debug	Debug-level messages

6.3 Output destinations for log messages

You can set the action that the SR Linux takes for input messages that meet the criteria specified in a filter. This can include sending the messages to a destination such as a log file, the console, or a remote host.

For example, you can configure the SR Linux to send messages generated by the kernel that have priority warning to a file called `/var/log/srlinux/file/kernel-warning`.

Messages generated by an SR Linux subsystem, such as the bgp_mgr or gnmi_server application, can be sent to specified destinations.

Actions for messages matching a filter can include the following:

- Send the messages to a specified file in the `/var/log/srlinux/file/` directory.
- Send the messages to a buffer. A buffer is similar to a file, but uses memory as storage and is not persistent across system reboots. Messages sent to a buffer are stored in the `/var/log/srlinux/buffer/` directory.
- Send the messages to the console; that is, the Linux device `/dev/console`, which may be assigned to a serial device in hardware.
- Send the messages to one or more remote servers. You can specify a network-instance where rsyslogd is run and which serves as the source for the messages.

6.4 Defining filters

Filters target specific messages or groups of log messages within the input message source. Filter criteria for log messages can include the following:

- Specific text within a message
- Prefix text at the beginning of a message
- Linux facility that generated the message
- Regular expression matching text within the message
- Syslog tag of the message

Example:

The following example shows a configuration that creates a filter that matches messages from the Linux facility kern that have a priority of warning or higher. See Table 4: Logging priorities for a list of logging priorities.

```
--{ * candidate shared default }--[  ]--
# info system logging
 system {
     logging {
         filter f1 {
                facility kern {
                    priority {
                        match-above warning [
                    }
                }
         }
     }
 }
```

Example

The following example creates a filter that matches messages from the Linux facility local6 (where SR Linux subsystem messages are logged by default) that have a priority of informational or higher and contain the text `accounting`. This filter can be used to match messages from the SR Linux accounting subsystem.

```
--{ * candidate shared default }--[  ]--
# info system logging
 system {
     logging {
         filter f2 {
                contains accounting {
                facility local6 {
                    priority {
                        match-above informational [
                    }
                }
         }
     }
 }
```

6.5 Logging destination configuration

You can configure the SR Linux to send logging information to the following destinations:

- A log file
- Memory buffer storage
- The console (`/dev/console`)
- One or more remote servers

6.5.1 Specifying a log file destination

The SR Linux can send log messages to a specified log file. By default, the log file resides in the `/var/log/srlinux/file` directory. You can specify the retention policy for the file, including the maximum size (default 10 Mb), as well as the number of files to keep in the rotation (default 4 files).

Example:

The following example uses messages from Linux facility cron as input, filters the messages for those that have critical priority, and sends the filtered messages to the file `/var/log/srlinux/file/cron-critical`:

```
--{ * candidate shared default }--[  ]--
# info system logging
 system {
     logging {
         file cron-critical {
                facility cron {
                    priority {
                        match-exact [
                            critical
                        ]
                    }
                }
         }
     }
 }
```

Example

The following example sends messages matching criteria specified in filter f1 to the file `/var/log/srlinux/file/f1-match`:

```
--{ * candidate shared default }--[  ]--
# info system logging
 system {
     logging {
         file f1-match {
                filter [
                    f1
                ]
         }
     }
 }
```

Example

The following example uses messages generated by the SR Linux AAA subsystem (that is, messages generated by the aaa_mgr application, but not including accounting messages) as input. The messages are filtered for those that have warning or informational priority, and the filtered messages are sent to the file `/var/log/srlinux/file/aaa-warn-info`.

```
--{ * candidate shared default }--[  ]--
# info system logging
 system {
     logging {
         file aaa-warn-info {
                subsystem aaa {
                    priority {
                        match-exact [
                            warning
```

```
                            informational
                        ]
                    }
                }
            }
        }
    }
```

6.5.2 Specifying a buffer destination

The SR Linux can send log messages to a buffer. A buffer is similar to a file, except that a buffer uses memory as storage and is not persistent across system reboots.

When the SR Linux device boots, it creates a non-swappable tmpfs virtual filesystem at `/var/log/srlinux/buffer`. This tmpfs filesystem has a fixed size of 512 Mb, which is reserved for buffer usage.

When a buffer is created through a commit transaction, the SR Linux verifies that there is enough buffer space available to contain all configured buffers based on their retention policies. If sufficient space is not available, the commit transaction fails.

Example:

The following example sends messages matching criteria specified in filter f1 to the buffer `/var/log/srlinux/buffer/f1-match`. A retention policy is specified so that when the buffer reaches 5,000,000 bytes, messages are written to a new buffer. After five buffers are filled, the oldest one is overwritten.

```
--{ * candidate shared default }--[  ]--
# info system logging
 system {
     logging {
         buffer f1-match {
                rotate 5
                size 5000000
                filter [
                    f1
                ]
         }
     }
 }
```

6.5.3 Specifying the console as destination

You can specify the console as a destination for log messages. The console refers to Linux device `/dev/console`, The console may be assigned to a serial device in hardware.

Example:

The following example uses messages generated by the SR Linux accounting subsystem as input, filters the messages for those that have informational priority or higher, and sends the filtered messages to `/dev/console`:

```
--{ * candidate shared default }--[  ]--
# info system logging
 system {
     logging {
         console {
                subsystem accounting {
```

```
                    priority {
                        match-above informational
                    }
                }
            }
        }
    }
```

6.5.4 Specifying a remote server destination

The SR Linux can send log messages to one or more remote servers. You can specify the network-instance that the SR Linux uses to contact the remote servers. The rsyslogd process is run within the specified network-instance.

Example:

The following example uses messages generated by the SR Linux BGP subsystem (that is, messages generated by the bgp_mgr application) as input, filters the messages for those that have alert priority or higher, and sends the filtered messages to a remote server. The messages are sourced from the mgmt network-instance.

```
--{ * candidate shared default }--[  ]--
# info system logging
 system {
     logging {
         network-instance mgmt
         remote-server 1.1.2.2 {
                subsystem bgp {
                    priority {
                        match-above alert
                    }
                }
            }
        }
    }
 }
```

6.6 Specifying a Linux syslog facility for SR Linux subsystem messages

All of the messages generated by SR Linux subsystems (see Table 3: SR Linux logging subsystem names) are logged to the same Linux syslog facility. This allows you to filter messages from all SR Linux subsystems by capturing logs from this facility.

By default, SR Linux subsystem messages are logged to the Linux syslog facility local6. You can optionally specify a different syslog facility. See Table 2: Linux syslog facilities for the syslog facilities.

Example:

The following example changes the Linux syslog facility where messages generated by SR Linux subsystems are logged from the default of local6 to local7:

```
--{ * candidate shared default }--[  ]--
# info system logging
 system {
     logging {
         subsystem-facility local7
     }
```

```
}
```

7 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.

7.1 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*

7.2 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
```

7.3 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 Network-instances.

You can configure ACL policies to filter IPv4 and IPv6 packets entering or leaving a subinterface. See Access control lists.

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.

7.3.1 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.

7.3.2 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`.

7.3.3 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 {
            dhcp-client true {
            }
        }
        ipv6 {
            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 {
            address 192.168.12.1/30 {
                primary
            }
            address 192.168.12.2/30 {
                primary
            }
            address 192.168.12.2/30 {
            }
        }
        ipv6 {
            address 2001:1::192:168:12:2/126 {
                primary
            }
            address 2001:1::192:168:12:3/126 {
            }
        }
    }
```

7.3.4 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
        ipv4 {
            dhcp-client true {
            }
        }
    }
    subinterface 2 {
        admin-state enable
        ipv4 {
            dhcp-client true {
            }
        }
        vlan {
            encap {
                single-tagged {
```

```
                    vlan-id 101
                }
            }
        }
    }
```

7.3.5 Bridged subinterface configuration

Bridged subinterfaces are associated with a mac-vrf network-instance. 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
      }
```

7.4 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.

7.4.1 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 {
                address 172.16.1.1/24 {
                }
            }
        }
    }
```

7.5 Displaying interface statistics

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-1/1
    interface ethernet-1/1 {
        admin-state enable
        mtu 9232
        loopback-mode false
        ifindex 54
        oper-state down
        oper-down-reason lower-layer-down
        last-change 2020-06-04T15:06:35.920Z
        vlan-tagging false
        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 0
            out-unicast-packets 0
            out-broadcast-packets 0
            out-multicast-packets 0
            out-error-packets 0
            carrier-transitions 0
        }
        traffic-rate {
            in-bps 0
            out-bps 0
        }
        transceiver {
            admin-state enable
            tx-laser true
            oper-state up
            ddm-events false
            forward-error-correction disabled
            form-factor QSFP28
            ethernet-pmd 100GBASE-SR4
            connector-type MPO-1x12
            vendor "AVAGO           "
           vendor-part-number "AFBR-89CDDZ-AL1 "
            vendor-revision 01
            serial-number "AF1937GN050     "
            date-code "190910  "
            fault-condition false
            temperature {
                latest-value 32
            }
            voltage {
                latest-value 3.2809
            }
            channel 1 {
                wavelength 850.00
                input-power {
                    latest-value -33.98
                }
                output-power {
                    latest-value 0.26
                }
                laser-bias-current {
                    latest-value 7.494
                }
            }
            channel 2 {
```

```
                wavelength 850.00
                input-power {
                    latest-value -40.00
                }
                output-power {
                    latest-value 0.05
                }
                laser-bias-current {
                    latest-value 7.494
                }
            }
            channel 3 {
                wavelength 850.00
                input-power {
                    latest-value -23.28
                }
                output-power {
                    latest-value 0.02
                }
                laser-bias-current {
                    latest-value 7.494
                }
            }
            channel 4 {
                wavelength 850.00
                input-power {
                    latest-value -40.00
                }
                output-power {
                    latest-value 0.24
                }
                laser-bias-current {
                    latest-value 7.494
                }
            }
        }
        ethernet {
            port-speed 100G
            hw-mac-address 68:AB:09:A2:71:B0
            flow-control {
                receive 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
            }
        }
        sflow {
```

```
            admin-state enable
        }
        qos {
            output {
                unicast-queue 0 {
                    queue-parameters {
                        peak-rate-bps 101370000000
                        strict-priority true
                    }
                }
                unicast-queue 1 {
                    queue-parameters {
                        peak-rate-bps 101370000000
                        strict-priority true
                    }
                }
                unicast-queue 2 {
                    queue-parameters {
                        peak-rate-bps 101370000000
                        strict-priority true
                    }
                }
                unicast-queue 3 {
                    queue-parameters {
                        peak-rate-bps 101370000000
                        strict-priority true
                    }
                }
                unicast-queue 4 {
                    queue-parameters {
                        peak-rate-bps 101370000000
                        strict-priority true
                    }
                }
                unicast-queue 5 {
                    queue-parameters {
                        peak-rate-bps 101370000000
                        strict-priority true
                    }
                }
                unicast-queue 6 {
                    queue-parameters {
                        peak-rate-bps 101370000000
                        strict-priority true
                    }
                }
                unicast-queue 7 {
                    queue-parameters {
                        peak-rate-bps 101370000000
                        strict-priority true
                    }
                }
            }
        }
        queue-statistics {
            unicast-queue 0 {
                virtual-output-queue 1 {
                }
                virtual-output-queue 2 {
                }
                virtual-output-queue 3 {
                }
                virtual-output-queue 4 {
                }
            }
            unicast-queue 1 {
```

```
                virtual-output-queue 1 {
                }
                virtual-output-queue 2 {
                }
                virtual-output-queue 3 {
                }
                virtual-output-queue 4 {
                }
            }
            unicast-queue 2 {
                virtual-output-queue 1 {
                }
                virtual-output-queue 2 {
                }
                virtual-output-queue 3 {
                }
                virtual-output-queue 4 {
                }
            }
            unicast-queue 3 {
                virtual-output-queue 1 {
                }
                virtual-output-queue 2 {
                }
                virtual-output-queue 3 {
                }
                virtual-output-queue 4 {
                }
            }
            unicast-queue 4 {
                virtual-output-queue 1 {
                }
                virtual-output-queue 2 {
                }
                virtual-output-queue 3 {
                }
                virtual-output-queue 4 {
                }
            }
            unicast-queue 5 {
                virtual-output-queue 1 {
                }
                virtual-output-queue 2 {
                }
                virtual-output-queue 3 {
                }
                virtual-output-queue 4 {
                }
            }
            unicast-queue 6 {
                virtual-output-queue 1 {
                }
                virtual-output-queue 2 {
                }
                virtual-output-queue 3 {
                }
                virtual-output-queue 4 {
                }
            }
            unicast-queue 7 {
                virtual-output-queue 1 {
                }
                virtual-output-queue 2 {
                }
                virtual-output-queue 3 {
                }
```

```
                virtual-output-queue 4 {
                }
            }
            multicast-queue 0 {
            }
            multicast-queue 1 {
            }
            multicast-queue 2 {
            }
            multicast-queue 3 {
            }
            multicast-queue 4 {
            }
            multicast-queue 5 {
            }
            multicast-queue 6 {
            }
            multicast-queue 7 {
            }
        }
    }
```

7.5.1 Clearing interface statistics

You can clear the statistics counters for a specified interface.

Example:

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

Example

To clear queue statistics for an interface:

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

Example

To clear statistics for a specified queue on an interface:

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

7.6 Displaying subinterface statistics

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

Example:

```
--{ candidate shared default }--[  ]--
# interface ethernet-1/2
--{ candidate shared default }--[ interface ethernet-1/2 ]--
# subinterface 1
--{ candidate shared default }--[ interface ethernet-1/2 subinterface 1 ]--
```

```
# info from state
    admin-state enable
    ip-mtu 1500
    ifindex 32769
    oper-state up
    last-change 2019-09-30T16:39:29.725Z
    ipv4 {
        allow-directed-broadcast false
        address 192.168.12.2/30 {
            origin static
        }
        arp {
            timeout 14400
            neighbor 192.168.12.1 {
                link-layer-address 00:01:01:FF:00:01
                origin dynamic
                expiration-time 2019-09-30T20:39:30.591Z
            }
        }
    }
    ipv6 {
        address 2001:1::192:168:12:2/126 {
            origin static
            status preferred
        }
        address fe80::201:3ff:feff:1/64 {
            origin link-layer
            status preferred
        }
        neighbor-discovery {
            dup-addr-detect true
            reachable-time 30
            stale-time 14400
            neighbor 2001:1::192:168:12:1 {
                link-layer-address 00:01:01:FF:00:01
                origin dynamic
                is-router true
                current-state reachable
                next-state-time 2019-09-30T17:26:30.018Z
            }
            neighbor fe80::201:1ff:feff:1 {
                link-layer-address 00:01:01:FF:00:01
                origin dynamic
                is-router true
                current-state stale
                next-state-time 2019-09-30T20:40:16.078Z
            }
        }
    }
    statistics {
        in-pkts 564
        in-octets 49394
        in-error-pkts 0
        in-discarded-pkts 0
        in-terminated-pkts 560
        in-terminated-octets 49054
        in-forwarded-pkts 4
        in-forwarded-octets 340
        out-forwarded-pkts 0
        out-forwarded-octets 0
        out-error-pkts 0
        out-discarded-pkts 0
        out-pkts 0
        out-octets 0
    }
    acl {
```

```
    }
    qos {
        input {
            classifiers {
                ipv4-dscp default
                ipv6-dscp default
                mpls-tc default
            }
        }
    }
```

```
vprn 1 customer 1 create
:
:
            ip-local-pool "pool-1"
                hold-timer 0
                ipv4-prefix "10.10.0.0/16"
            exit
            ip-local-pool "pool-2"
                hold-timer 0
                ipv4-prefix "10.11.0.0/16"
                ipv4-prefix "10.12.0.0/16"
            exit
            static-route 10.10.0.0/16 black-hole
            static-route 10.11.0.0/16 black-hole
            static-route 10.12.0.0/16 black-hole
:
:
            no shutdown
exit
```

```
vprn 2 customer 1 create
:
:
            ip-local-pool "pool-3"
                hold-timer 0
                ipv4-prefix "11.10.0.0/16"
            exit
            ip-local-pool "pool-4"
                hold-timer 0
                ipv4-prefix "1.11.0.0/16"
                ipv4-prefix "11.12.0.0/16"
            exit
            static-route 11.10.0.0/16 black-hole
            static-route 11.11.0.0/16 black-hole
            static-route 11.12.0.0/16 black-hole
:
:
            no shutdown
exit
```

7.6.1 Clearing subinterface statistics

You can clear the statistics counters for a specified subinterface.

Example:

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

```
    subinterface ethernet-1/1.1 statistics cleared
```

7.7 Displaying interface status

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

Example:

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

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

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

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

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

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

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
===============================================================================
```

7.8 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.

7.8.1 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.

7.8.2 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.

7.8.2.1 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.

7.8.3 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
        }
    }
```

7.8.3.1 Configuring the min-link threshold

The min-link threshold 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).

7.8.3.2 Configuring LACP and LACP fallback

When LACP is enabled, 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:

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
 interface lag1 {
```

```
        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

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 }--[ ]--
 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

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 which can be influenced using LACP port priority.

```
 interface ethernet-1/1 {
        ethernet {
            aggregate-id lag1
            lacp-port-priority 1
            port-speed 25G
            hw-mac-address 00:01:02:FF:00:01
            statistics {
```

7.8.4 Displaying LAG interface statistics

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:

```
# 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
        }
    }
```

7.8.4.1 Clearing LAG interface statistics

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

Example:

```
# tools interface lag1 statistics clear
/interface[name=lag1]:
    interface lag1 statistics cleared
```

Example

To clear statistics for a LAG interface and all member links:

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

7.9 Breakout ports (7220 IXR-D3 only)

On 7220 IXR-D3 systems, the QSFP28 connector ports (ports 1/3-1/33) can operate in breakout mode. Each QSFP28 connector port operating in breakout mode can have four breakout ports configured, each operating at 25G.

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

7.9.1 Configuring breakout mode for an interface

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

Example:

```
--{ candidate shared default }--[  ]--
# info interface ethernet-1/3*
    interface ethernet-1/3 {
        admin-state enable
        description "Breakout connector"
        breakout-mode {
            num-channels 4
            channel-speed 25G
            }
        }
    }
    interface ethernet-1/3/1 {
        admin-state enable
        description "Breakout port 1"
        subinterface 1 {
            admin-state enable
            ipv4 {
                address 192.168.12.1/30 {
                }
            }
        }
    }
    interface ethernet-1/3/2 {
        admin-state enable
        description "Breakout port 2"
        subinterface 1 {
            admin-state enable
            ipv4 {
                address 192.168.12.2/30 {
                }
            }
        }
    }
    interface ethernet-1/3/3 {
        admin-state enable
        description "Breakout port 3"
        subinterface 1 {
            admin-state enable
            ipv4 {
                address 192.168.12.3/30 {
                }
            }
        }
    }
    interface ethernet-1/3/4 {
        admin-state enable
        description "Breakout port 4"
        subinterface 1 {
            admin-state enable
            ipv4 {
                address 192.168.12.4/30 {
                }
            }
        }
    }
```

7.10 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 1: 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 2: 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.

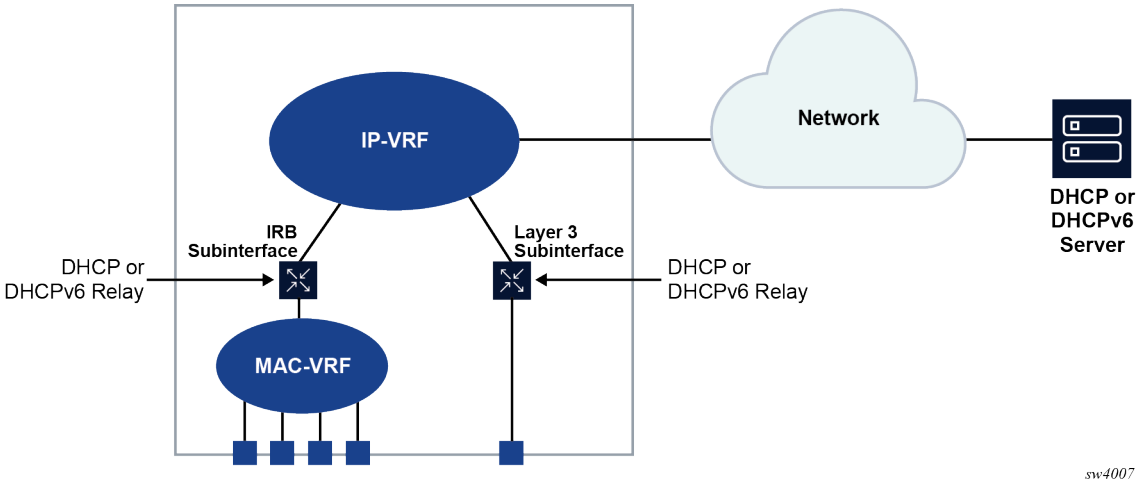

Figure 1: DHCP relay for IRB and Layer 3 subinterfaces

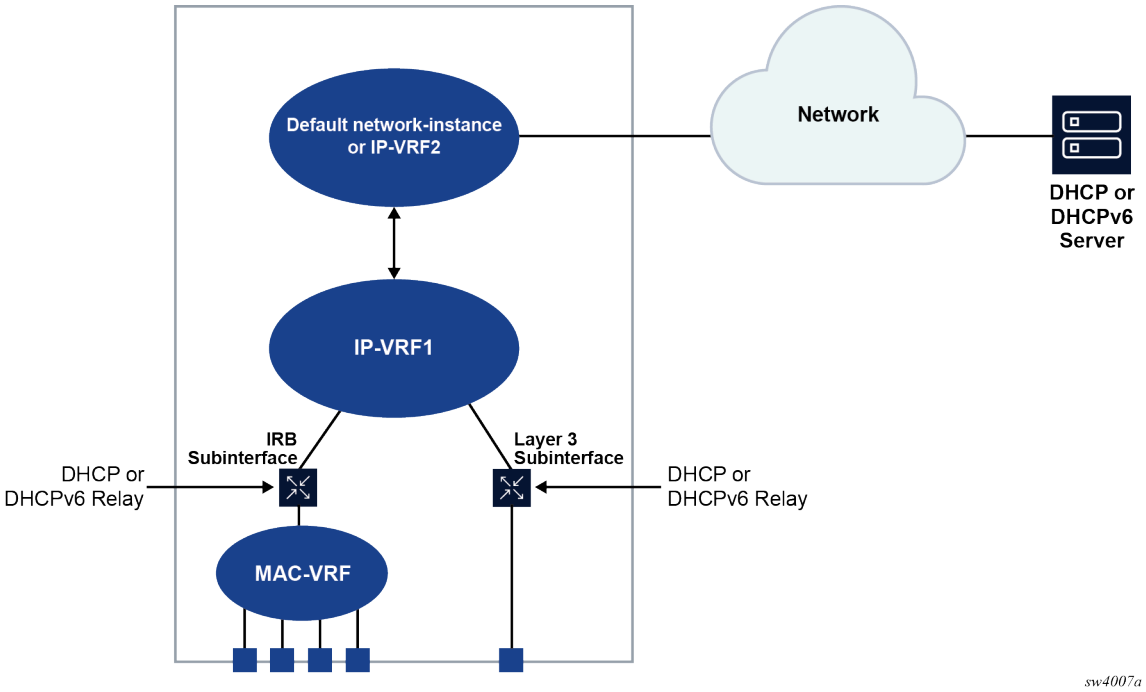

Figure 2: DHCP relay using different IP-VRF or default network-instance

7.10.1 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 3: 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.

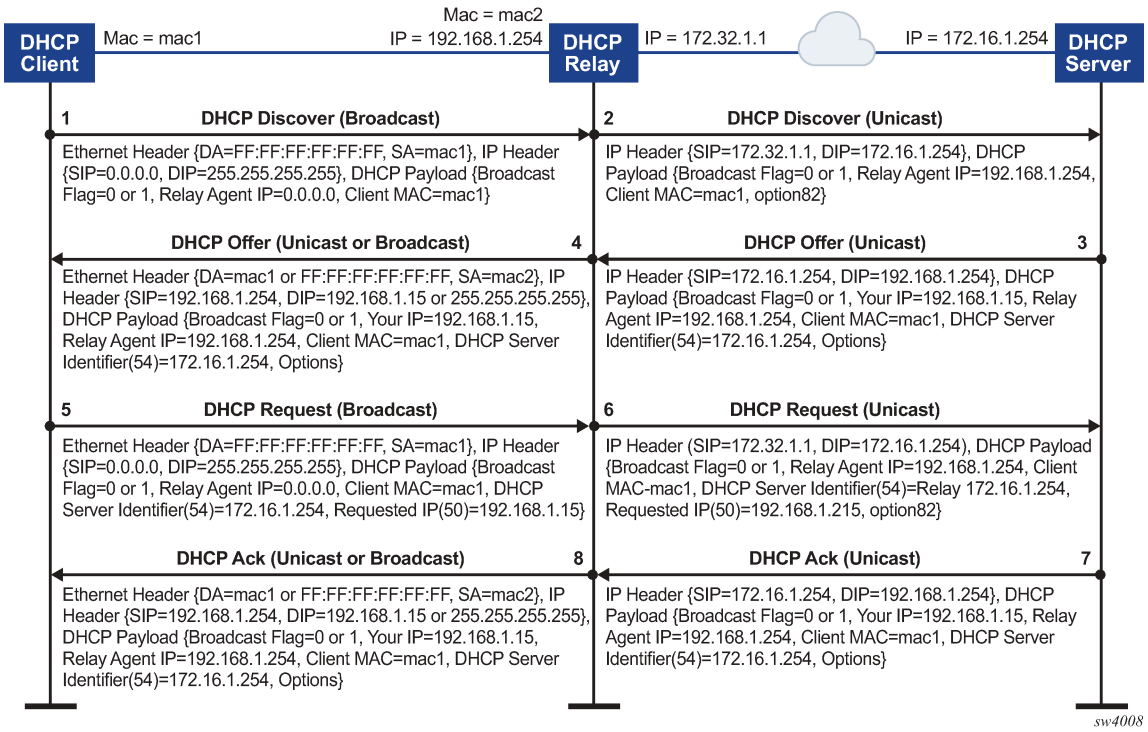

Figure 3: DHCP message flow for IPv4 address allocation

The DORA message flow shown in Figure 3: 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 DCHP 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.

7.10.1.1 Configuring DHCP relay for IPv4

To configure DHCP relay for a subinterface:

- Configure the addresses 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:

The following example configures the DHCP relay agent on a subinterface. The example configures the IP addresses 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 {
                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
                    ]
                }
            }
```

Example:

If the DHCP server network is in a different IP-VRF network-instance from the Layer 3 subinterfaces that require DHCP relay (see Figure 2: 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 {
                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
                    ]
                }
            }
```

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

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 5: 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
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 {
            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
                ]
            }
        }
```

7.10.1.3 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.

7.10.2 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.

Figure 4: DHCPv6 message flow for IPv6 address allocation shows the DHCPv6 message flow. Figure 5: DHCPv6 renew message flow and Figure 6: DHCPv6 release message flow show the renew and release flows.

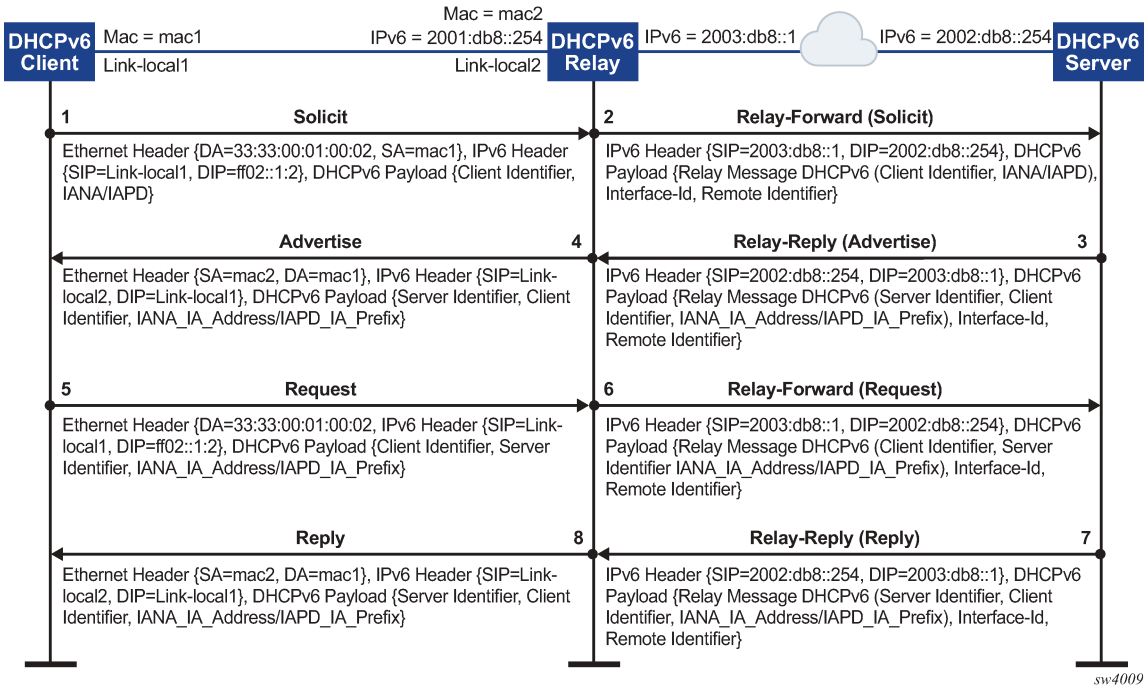

Figure 4: DHCPv6 message flow for IPv6 address allocation

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 DCHP 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).

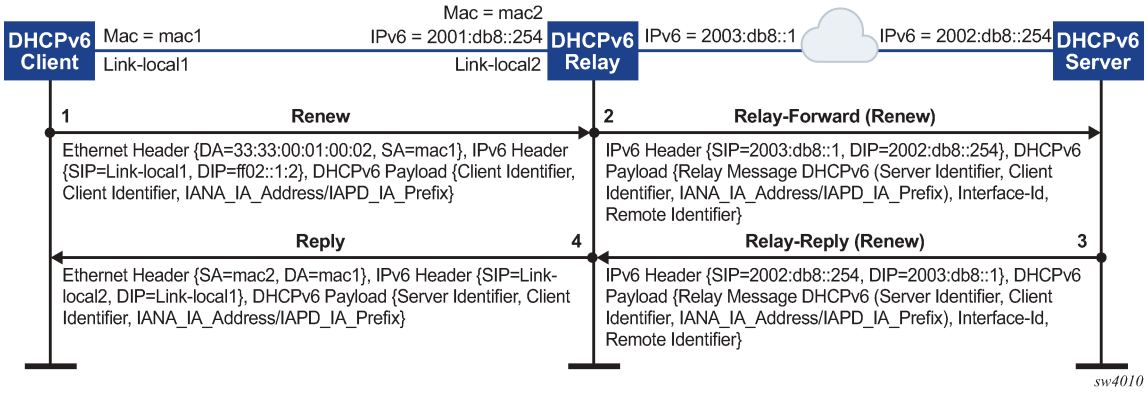

Figure 5: DHCPv6 renew message flow

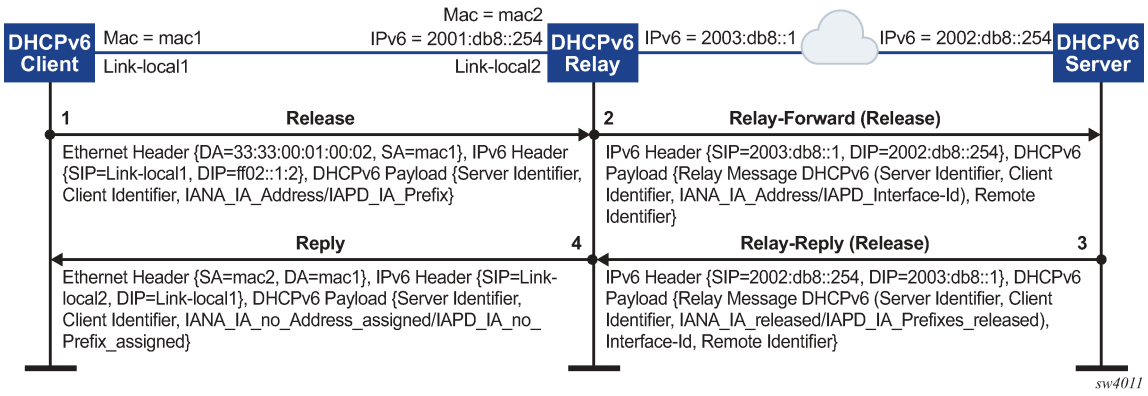

Figure 6: DHCPv6 release message flow

7.10.2.1 Configuring DHCP relay for IPv6

To configure DHCP relay for a subinterface for IPv6:

- Configure the addresses of the DHCPv6 servers.
- Optionally configure the source IPv6 address for DHCP messages sent to the servers.
- Configure whether information is included in the Interface-Id (option 18) and Remote Identifier (option 37).

Example:

The following example configures the DHCPv6 relay agent on a subinterface. The example configures the IP addresses of the remote DHCPv6 servers and specifies the address to be used as the source IPv6 address 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 Interface-Id (option 18) DHCPv6 client MAC address in the Remote Identifier (option 37).

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/2
 interface ethernet-1/2 {
        description dut1-dut4-1
        subinterface 1 {
            ipv6 {
                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
                    ]
                    server [
                       1::1
                       2::2
                    ]
                }
            }
```

```
       }
```

Example

If the DHCP server network is in a different IP-VRF network-instance from the Layer 3 subinterfaces that require DHCP relay (see Figure 2: 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 {
                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
                    ]
                    server [
                       1::1
                       2::2
                    ]
                }
            }
       }
```

7.10.3 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).

7.10.4 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.

7.10.5 Displaying DHCP relay statistics

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

```
# 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

```
# 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
                    }
                }
            }
        }
    }
```

7.10.5.1 Clearing DHCP relay statistics

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

Example:

```
# 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
```

7.11 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.

7.11.1 Configuring the DHCP server

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:

The following example enables 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 {
                address 192.14.1.4/27 {
                }
                dhcp-server {
                    admin-state enable
                }
            }
            ipv6 {
                address 2001:192:14:1::4/120 {
                }
                dhcpv6-server {
                    admin-state enable
                }
            }
        }
    }
```

Example

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

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

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
                                ]
                            }
                        }
                    }
                }
            }
        }
    }
```

7.12 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.

7.12.1 Configuring IPv6 router advertisements

The following example configures the 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 {
                router-advertisement {
                    router-role {
                        admin-state enable
                        prefix 2001:db8:0:b::/64 {
                        }
                    }
                }
            }
        }
    }
```

7.13 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.

7.13.1 Configuring IPv6 RA guard policies

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

Example:

The following example configures and 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

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
        }
    }
```

7.13.2 Applying IPv6 RA guard policies to subinterfaces

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

Example:

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

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 {
                }
            }
        }
    }
```

7.14 Interface port speed configuration

By default, ports on SR Linux devices operate at the speeds listed in Table 6: 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 6: 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
	Ports 3-34	400G	40G, 100G, 400G

7.14.1 Configuring interface port speed

The following example configures the port speed for an interface.

Example:

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

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

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 7: 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 7: 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 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 from state interface ethernet-1/1
    interface ethernet-1/1 {
        ethernet {
            auto-negotiate true
            port-speed 100M
            }
```

```
        }
    }
```

8 Network-instances

On the SR Linux device, you can configure one or more virtual routing instances, known as "network-instances". Each network-instance has its own interfaces, its own protocol instances, its own route table, and its own FIB.

When a packet arrives on a subinterface associated with a network-instance, it is forwarded according to the FIB of that network-instance. Transit packets are normally forwarded out another subinterface of the network-instance.

The SR Linux supports three types of network-instances: **default**, **ip-vrf**, and **mac-vrf**. Type **default** is the default network-instance and only one of this type is supported. Type **ip-vrf** is the regular network-instance; you can create multiple network-instances of this type.

Type **mac-vrf** functions as a broadcast domain and is associated with an **ip-vrf** network-instance via an Integrated Routing and Bridging (IRB) to support tunneling of Layer 2 traffic across an IP network. See mac-vrf network-instance.

Initially, the SR Linux has a default network-instance and no ip-vrf or mac-vrf network-instances.

A management network-instance, which isolates management network traffic from other network-instances configured on the device, is created by default, with the `mgmt0` port automatically added to it. See Configuring the management network-instance and Interfaces.

8.1 Basic network-instance configuration

The following example creates network-instance "black" and associates two network subinterfaces and one loopback subinterface with it.

The configuration administratively enables the network-instance, specifies a description, and assigns it a router ID. The router ID is optional and is used as a default router identifier for protocols running within the network-instance.

The network-instance is configured to export routes and neighbors to the Linux routing table.

```
--{ candidate shared default }--[  ]--
# info network-instance black
 network-instance black {
        description "Sample network instance"
        type ip-vrf
        admin-state enable
        router-id 1.1.1.1
        interface ethernet-1/1.1 {
        }
        interface ethernet-1/2.1 {
        }
        interface lo0.1 {
        }
        protocols {
            linux {
                export-routes true
                export-neighbors true
            }
        ip-forwarding {
                receive-ipv4-check true
```

```
        }
 }
```

In the preceding example, the **receive-ipv4-check** parameter is set to **true**; if an IPv4 packet is received on a subinterface of this network-instance, and the IPv4 operational status of the subinterface is down, then the packet is discarded. When the **receive-ipv4-check** parameter is set to **false**, IPv4 packets are received on all subinterfaces of this network-instance that are up, even if they do not have IPv4 addresses.

8.2 Path MTU discovery

Path MTU discovery is the technique for determining the MTU size on the network path between hosts. On the SR Linux, path MTU discovery is enabled by default for all network-instances, and can be manually enabled or disabled per network-instance.

If path MTU discovery is disabled, the system drops the MTU for the session to the number of bytes specified by the **min-path-mtu** parameter when an ICMP fragmentation-needed message is received; by default the **min-path-mtu** setting is 552 bytes.

```
--{ * candidate shared default }--[  ]--
# info network-instance black
   network-instance black  {
     mtu {
       path-mtu-discovery true
       min-path-mtu 552
      }
   }
```

8.3 Static routes

Within a network-instance, you can configure static routes. Each static route is associated with an IPv4 prefix or an IPv6 prefix, which represents the packet destinations matched by the static route. Each static route belongs to a specific network-instance. Different network-instances can have overlapping routes (static or otherwise) because each network-instance installs its own routes into its own set of route tables and FIBs.

Each static route must be associated with a statically configured next-hop group, which determines how matching packets are handled: either perform a blackhole discard action or a forwarding action. The next-hop group can specify a list of one or more next-hops (up to 128), each identified by an IPv4 or IPv6 address and a resolve flag. If the resolve flag is set to false, only a direct route can be used to resolve the IPv4 or IPv6 next-hop address; if the resolve flag is set to true, any route in the FIB can be used to resolve the IPv4 or IPv6 next-hop address.

Each static route has a specified metric and preference. The metric is the IGP cost to reach the destination. The preference specifies the relative degree this static route is preferred compared to other static and non-static routes available for the same IP prefix in the same network-instance.

A static route is installed in the FIB for the network-instance if the following conditions are met:

- The route has the lowest preference value among all routes (static and non-static) for the IP prefix.
- The route has the lowest metric value among all static routes for the IP prefix.

If BGP is running in a network-instance, all static routes of that same network-instance are automatically imported into the BGP local RIB, so that they can be redistributed as BGP routes, subject to BGP export policies.

You can use BFD to monitor reachability between the router and the configured next hops for a static route, making BFD sessions between the local router and the defined next hops a condition for an associated static route and next hops to be operationally active. See Configuring failure detection for static routes.

8.3.1 Configuring static routes

To configure static routes, you specify route prefixes to point to next-hop groups, along with the metric and preference.

Example:

The following example configures IPv4 and IPv6 static route prefixes to point to next-hop groups and specifies a preference and metric for each one:

```
--{ * candidate shared default }--[ network-instance black ]--
# info static-routes
        static-routes {
            route 192.168.18.0/24 {
                admin-state enable
                metric 1
                preference 5
                next-hop-group static-ipv4-grp
            }
            route 2001:1::192:168:18:0/64 {
                admin-state enable
                metric 1
                preference 6
                next-hop-group static-ipv6-grp
            }
        }
```

Example

The following example configures the next-hop groups for the static routes:

```
--{ * candidate shared default }--[ network-instance black ]--
# info next-hop-groups
        next-hop-groups {
            group static-ipv4-grp {
                admin-state enable
                nexthop 1 {
                    ip-address 172.16.0.22
                }
                nexthop 2 {
                    ip-address 172.16.0.45
                    resolve true
                }
            }
            group static-ipv6-grp {
                admin-state enable
                blackhole {
                }
            }
        }
    }
```

In the preceding example, an IPv4 next-hop group is configured with two next-hops. The `resolve true` setting allows any route in the FIB to be used to resolve the IPv4 next-hop address, provided the resolution depth is not more than 2.

The IPv6 next-hop group is configured to perform a blackhole discard action for matching packets.

8.3.2 Configuring failure detection for static routes

BFD can be used as a failure detection mechanism for monitoring the reachability of next hops for static routes. When BFD is enabled for a static route, it makes an active BFD session between the local router and the defined next hops required as a condition for a static route to be operationally active.

BFD is enabled for specific next-hop groups; as a result, BFD is enabled for any static route that refers to the next-hop group. If multiple next hops are defined within the next-hop group, a BFD session is established between the local address and each next hop in the next-hop group.

A static route is considered operationally up if at least one of the configured next-hop addresses can establish a BFD session. If the BFD session fails, the associated next hop is removed from the FIB as an active next hop.

Example:

The following example enables BFD for a static route next-hop:

```
--{ * candidate shared default }--[ network-instance black ]--
# info next-hop-groups
    next-hop-groups {
        group static-ipv4-grp {
            admin-state enable
            nexthop 1 {
                failure-detection {
                    enable-bfd {
                        local-address 1.2.34.5
                    }
                }
            }
        }
    }
```

A BFD session is established between the address configured with the **local-address** parameter and each next-hop address before that next-hop address is installed in the forwarding table.

All next-hop BFD sessions share the same timer settings, which are taken from the BFD configuration for the subinterface where the address in **local-address** parameter is configured. See BFD.

8.4 Aggregate routes

You can specify aggregate routes for a network-instance. Each aggregate route is associated with an IPv4 prefix or an IPv6 prefix, which represents the packet destinations matched by the aggregate route. As with static routes, each aggregate route belongs to a specific network-instance, though different network-instances can have overlapping routes because each network-instance installs its own routes into its own set of route tables and FIBs.

An aggregate route can become active when it has one or more contributing routes. A route contributes to an aggregate route if all of the following conditions are met:

- The prefix length of the contributing route is greater than the prefix length of the aggregate route.

- The prefix bits of the contributing route match the prefix bits of the aggregate route up to the prefix length of the aggregate route.
- There is no other aggregate route that has a longer prefix length that meets the previous two conditions.
- The contributing route is actively used for forwarding and is not an aggregate route itself.

That is, a route can only contribute to a single aggregate route, and that aggregate route cannot recursively contribute to a less-specific aggregate route.

Aggregate routes have a fixed preference value of 130. If there is no route to the aggregate route prefix with a numerically lower preference value, then the aggregate route, when activated by a contributing route, is installed into the FIB with a blackhole next-hop. It is not possible to install an aggregate route into the route-table or as a BGP route without also installing it in the FIB.

The aggregate routes are commonly advertised by BGP or another routing protocol so that the individual contributing routes no longer need to be advertised.

This can speed up routing convergence and reduce RIB and FIB sizes throughout the network. If BGP is running in a network-instance, all active aggregate routes of that network-instance are automatically imported into the BGP local RIB so they can be redistributed as BGP routes, subject to BGP export policies.

8.4.1 Configuring aggregate routes

The following example specifies an aggregate route. The aggregator address setting identifies the aggregating router. By default, this is the configured router ID of the BGP instance, or 0 if BGP is not enabled.

Example:

```
--{ * candidate shared default }--[ network-instance black ]--
# info aggregate-routes
        aggregate-routes {
            route 172.16.0.0/24 {
                aggregator {
                    address 1.1.1.1
                }
                summary-only true
                generate-icmp true
            }
        }
```

When the **summary-only** parameter is set to **true**, activation of an aggregate route automatically blocks the advertisement of all of its contributing routes by BGP.

The **generate-icmp true** setting causes the router to generate ICMP unreachable messages for the dropped packets.

8.5 Route preferences

A route can be learned by the router from different protocols, in which case, the costs are not comparable. When a route is learned from different protocols, the preference value is used to decide which route is installed in the forwarding table if several protocols calculate routes to the same destination. The route with the lowest preference value is selected.

Different protocols should not be configured with the same preference. If protocols are configured with the same preference, the tiebreaker is per the default preference table as defined in Table 8: Route preference defaults by route type. If multiple routes are learned with an identical preference using the same protocol, the lowest cost route is used.

Table 8: Route preference defaults by route type

Route Type	Preference	Configurable
Direct attached	0	No
Static routes	5	Yes
OSPF internal	10	Yes[1]
IS-IS level 1 internal	15	Yes
IS-IS level 2 internal	18	Yes
OSPF external	150	Yes
IS-IS level 1 external	160	Yes
IS-IS level 2 external	165	Yes
BGP	170	Yes

Note:

1. Preference for OSPF internal routes is configured with the preference command.

8.6 Displaying network-instance status

Use the **show network-instance** command to display status information about network-instances configured on the device.

Example:

To display information about all configured network-instances, including the router ID, description, administrative and operational state:

```
--{ show }--
# show network-instance summary
+---------+---------+--------+-------+------------+--------------------------------+
|  Name   |  Type   | Admin  | Oper  | Router id  |   Description                  |
|         |         | state  | state |            |                                |
+=========+=========+========+=======+============+================================+
| default | default | enable | up    | 5.5.5.5    | Sample network instance        |
| mgmt    | ip-vrf  | enable | up    |            | Management network instance    |
| red     | ip-vrf  | enable | up    |55.55.55.55 | Network instance for bgp tests |
+---------+---------+--------+-------+------------+--------------------------------+
```

Example

To limit the display to a single network-instance:

```
--{ show }--
# show network-instance default summary
+---------+---------+--------+-------+---------+---------------------------+
|  Name   |  Type   | Admin  | Oper  | Router  |         Description       |
|         |         | state  | state |   id    |                           |
+=========+=========+========+=======+=========+===========================+
| default | default | enable | up    | 5.5.5.5 | "Sample network instance"  |
+---------+---------+--------+-------+---------+---------------------------+
```

Example

To display information about the interfaces attached to a network-instance:

```
--{ show }--
# show network-instance default interfaces
=============================================================
Net instance    : default
Interface       : ethernet-1/1.1
Oper state      : up
Ip mtu          : 1500
                  Prefix                    Origin        Status
  ===========================================================
  192.35.1.0/31                           static
  2001:192:35:1::/127                     static       preferred
  fe80::201:5ff:feff:0/64                 link-layer   preferred
=============================================================
Net instance    : default
Interface       : lo0.1
Oper state      : up
                  Prefix                    Origin        Status
  ===========================================================
  5.5.5.5/32                              static
  2001:5:5:5::5/128                       static       preferred
=============================================================
```

The command displays the operational state, IP MTU, and assigned IPv4/IPv6 prefix for each interface. If the operational state for an interface is down, the reason for the interface being down is shown.

8.7 mac-vrf network-instance

The network instance type **mac-vrf** is associated with a network-instance of type **default** or **ip-vrf** via an Integrated Routing and Bridging (IRB) interface.

The mac-vrf network-instance type functions as a broadcast domain. Each mac-vrf network-instance builds a bridge table composed of MAC addresses that can be learned via the data path on network-instance interfaces or via static configuration. You can configure the size of the bridge table for each mac-vrf network instance, as well as the aging for dynamically learned MAC addresses and other parameters related to the bridge table.

The mac-vrf network-instance type features a mac duplication mechanism that monitors MAC address moves across network-instance interfaces and across interfaces.

8.7.1 MAC selection

Each mac-vrf network-instance builds a bridge table to forward Layer 2 frames based on a MAC address lookup. The SR Linux selects the MAC addresses to be sent for installation to the line card (XDP), based on the following priority:

1. Local application MACs
2. Local static MACs
3. EVPN static (MACs coming from a MAC/IP route with the static bit set)
4. Local duplicate MACs
5. Learned or EVPN-learned MACs

8.7.2 MAC duplication detection and actions

MAC duplication is the mechanism used by SR Linux for loop prevention. MAC duplication monitors MAC addresses that move between subinterfaces. It consists of detection, actions, and process restarts.

8.7.2.1 MAC duplication detection

Detection of duplicate MAC addresses is necessary when extending broadcast domains to multiple leaf nodes. SR Linux supports a MAC duplication mechanism that monitors MAC address moves across network-instance interfaces.

A MAC address is considered a duplicate when its number of detected moves is greater than a configured threshold within a configured time frame where the moves are observed. Upon exceeding the threshold, the system holds on to the prior local destination of the MAC and executes an action.

8.7.2.2 MAC duplication actions

The action taken upon detecting one or more MAC addresses as duplicate on a subinterface can be configured for the mac-vrf network instance or for the subinterface. The following are the configurable actions:

- **oper-down** – When one or more duplicate MAC addresses are detected on the subinterface, the subinterface is brought operationally down.
- **blackhole** – upon detecting a duplicate mac on the subinterface, the mac will be blackholed.
- **stop learning** – Upon detecting a duplicate mac on the subinterface, the MAC address will not be relearned anymore on this or any subinterface. This is the default action for a mac-vrf network instance.
- **use-network-instance-action** – (Available for subinterfaces only) Use the action specified for the mac-vrf network instance. This is the default action for a subinterface.

8.7.2.3 MAC duplication process restarts

When at least one duplicate MAC address is detected, the duplicate MAC addresses are visible in the state datastore and can be displayed with the **info from state mac-duplication duplicate-entries** CLI command.

8.7.2.4 Configurable hold-down-time

The **info from state mac-duplication duplicate-entries** command also displays the hold-down-time for each duplicate MAC address. When the hold-down-time expires for all of the duplicate MAC addresses for the subinterface, the oper-down or stop-learning action is cleared, and the subinterface is brought operationally up or starts learning again.

The hold-down-time is configurable from between 2 and 60 minutes. You can optionally specify **indefinite** for the hold-down-time, which prevents the oper-down or stop-learning action from being cleared after a duplicate MAC address is detected; in this case, you can manually clear the oper-down or stop-learning action by changing the mac-duplication configuration or using the **tools network-instance bridge-table mac-duplication** command.

8.7.3 Bridge table configuration

The bridge table, its MAC address limit, and maximum number of entries can be configured on a per mac-vrf or per-subinterface basis.

When the size of the bridge table exceeds its maximum number of entries, the MAC addresses are removed in reverse order of the priority listed in MAC selection.

You can also configure aging for dynamically learned MAC addresses and other parameters related to the bridge table.

8.7.3.1 Delete entries from the bridge table

The SR Linux features commands to delete duplicate or learned MAC entries from the bridge table. For a mac-vrf or subinterface, you can delete all MAC entries, MAC entries with a blackhole destination, or a specific MAC entry.

Example:

The following example clears MAC entries in the bridge table for a mac-vrf network instance that have a blackhole destination:

```
--{ candidate shared default }--[  ]--
# tools network-instance mac-vrf-1 bridge-table mac-duplication delete-blackhole-macs
```

Example

The following example deletes a specified learned MAC address from the bridge table for a mac-vrf network instance:

```
--{ candidate shared default }--[  ]--
# tools network-instance mac-vrf-1 bridge-table mac-learning delete-mac 00:00:00:00:00:04
```

Example

The following example clears all duplicate MAC entries in the bridge table for a subinterface:

```
--{ candidate shared default }--[  ]--
# tools interface ethernet-1/1.1 bridge-table mac-duplication delete-all-macs
```

8.7.4 mac-vrf network instance configuration

The following example configures a mac-vrf network instance and settings for the bridge table. The bridge table is set to a maximum of 500 entries. Learned MAC addresses are aged out of the bridge table after 600 seconds.

MAC duplication detection is configured so that a MAC address is considered a duplicate when its number of detected moves across network instance interfaces is greater than 3 over a 5-minute interval. In this example, the MAC address is blackholed. After the hold-down-time of 3 minutes, the MAC address is flushed from the bridge table, and the monitoring process for the MAC address is restarted.

The example includes configuration for a static MAC address in the bridge table.

The mac-vrf network instance is associated with a bridged interface and an IRB interface.

```
--{ candidate shared default }--[  ]--
# info network-instance mac-vrf-1
 network-instance mac-vrf-1 {
        description "Sample mac-vrf network instance"
        type mac-vrf
        admin-state enable
        interface ethernet-1/1.1 {
        }
        interface irb1.1 {
        }
        bridge-table {
            mac-limit {
                mac-limit 500
            }
            mac-learning {
                admin-state enable
                aging {
                    admin-state enable
                    age-time 600
                }
            }
            mac-duplication {
                admin-state enable
                monitoring-window 5
                num-moves 3
                hold-down-time 3
                action blackhole
            static-mac {
                address [mac1
                }
            }
            }
        }
 }
```

9 OSPF

Open Shortest Path First (OSPF) is a hierarchical link state protocol. OSPF is an interior gateway protocol (IGP) used within large autonomous systems (ASs). OSPF routers exchange state, cost, and other relevant interface information with neighbors. The information exchange enables all participating routers to establish a network topology map. Each router applies the Dijkstra algorithm to calculate the shortest path to each destination in the network. The resulting OSPF forwarding table is submitted to the routing table manager to calculate the routing table.

When a router is started with OSPF configured, OSPF, along with the routing-protocol data structures, is initialized and waits for indications from lower-layer protocols that its interfaces are functional.

The hierarchical design of OSPF allows a collection of networks to be grouped into a logical area. An area's topology is concealed from the rest of the AS which significantly reduces OSPF protocol traffic. With the correct network design and area route aggregation, the size of the route table can be greatly reduced, resulting in decreased OSPF route calculation time and topological database size.

Routers that belong to more than one area are called area border routers (ABRs). An ABR maintains a separate topological database for each area it is connected to. Every router that belongs to the same area has an identical topological database for that area.

Key OSPF areas are:

- Backbone Areas – The backbone distributes routing information between areas.
- Stub areas – A designated area that does not allow external route advertisements. Routers in a stub area do not maintain external routes. A single default route to an ABR replaces all external routes.
- Not-So-Stubby Areas (NSSAs) – NSSAs are similar to stub areas in that no external routes are imported into the area from other OSPF areas. External routes learned by OSPF routers in the NSSA area are advertised as type-7 LSAs within the NSSA area and are translated by ABRs into type-5 external route advertisements for distribution into other areas of the OSPF domain.

9.1 OSPF global configuration

The minimal OSPF parameters that should be configured to deploy OSPF are:

- OSPF version

  SR Linux supports OSPF version 2 and version 3. The OSPF version number must be specified in the configuration. If configuring OSPFv3, you must also specify the address family to be used, either IPv4 or IPv6.

- OSPF instance ID (when configuring multiple instances)

  An OSPF instance ID must be defined when configuring multiple instances or the instance being configured is not the base instance. If an instance ID is not configured, the default instance IDs are as follows:

  – 0 for OSPFv2

  – 0 for OSPF v3 with address family IPv6 unicast

  – 64 for OSPF v3 with address family IPv4 unicast

- Router ID

  Each router running OSPF must be configured with a unique router ID. The router ID is used by both OSPF and BGP routing protocols in the routing table manager.

  When you configure a new router ID, OSPF is automatically disabled and re-enabled to initialize the new router ID.

- An area

  At least one OSPF area must be created. An interface must be assigned to each OSPF area.

- Interfaces

  An interface is the connection between a router and one of its attached networks. An interface has state information associated with it, which is obtained from the underlying lower level protocols and the routing protocol itself. An interface to a network has associated with it a single IP address and mask (unless the network is an unnumbered point-to-point network). An interface is sometimes also referred to as a link.

9.1.1 Configuring basic OSPF parameters

To create a basic OSPF configuration, the minimal OSPF parameters required are the following:

- OSPF version number, either v2 or v3. If configuring OSPFv3, you must specify the address family, either IPv4 or IPv6.
- A router ID
- One or more areas
- Interfaces

Example:

The following is an example of a basic OSPFv2 configuration:

```
--{ * candidate shared default }--[ network-instance default protocols ]--
# info ospf
    ospf {
        instance default {
            admin-state enable
            version ospf-v2
            router-id 1.1.1.1
            area 0.0.0.1 {
                interface ethernet-1/1.1 {
                    interface-type broadcast
                }
                interface ethernet-1/2.1 {
                    interface-type broadcast
                }
                interface ethernet-1/16.1 {
                    interface-type broadcast
                }
                interface lo0.1 {
                    interface-type broadcast
                }
            }
        }
    }
```

Example

The following is an example of a basic OSPFv3 configuration.

```
--{ * candidate shared default }--[ network-instance default protocols ]--
# info ospf
    ospf {
        instance default {
            admin-state enable
            version ospf-v3
            address-family ipv6-unicast
            router-id 1.1.1.1
            area 0.0.0.1 {
                interface ethernet-1/1.1 {
                    interface-type broadcast
                }
                interface ethernet-1/2.1 {
                    interface-type broadcast
                }
                interface ethernet-1/16.1 {
                    interface-type broadcast
                }
                interface lo0.1 {
                    interface-type broadcast
                }
            }
        }
    }
```

9.1.2 Configuring the router ID

The router ID, expressed like an IP address, uniquely identifies the router within an AS. In OSPF, routing information is exchanged between ASs, groups of networks that share routing information. It can be set to be the same as the loopback (system interface) address. Subscriber services also use this address as far-end router identifiers when service distribution paths (SDPs) are created. The router ID is used by both OSPF and BGP routing protocols.

When you configure a new router ID, OSPF is automatically disabled and re-enabled to initialize the new router ID.

The router ID can be configured either at the network-instance level or at the OSPF protocol level. If a router ID is configured at the OSPF protocol level, OSPF uses it instead of the router ID configured at the network-instance level.

Example:

The following example configures a router ID at the network-instance level. OSPF uses this router ID unless a different router ID is configured at the OSPF protocol level.

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        router-id 10.10.10.104{
        }
    }
```

Example

The following example configures a router ID for the OSPF instance at the OSPF protocol level:

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            ospf {
                instance default {
                    version ospf-v2
                    router-id 2.2.2.2
                }
            }
        }
    }
```

9.1.3 Configuring an area

An OSPF area consists of routers configured with the same area ID. To include a router in a specific area, the common area ID must be assigned and an interface identified.

If your network consists of multiple areas, you must also configure a backbone area (0.0.0.0) on at least one router. The backbone comprises the area border routers and other routers not included in other areas. The backbone distributes routing information between areas. The backbone is considered to be a participating area within the autonomous system. To maintain backbone connectivity, there must be at least one interface in the backbone area.

The minimal configuration must include an area ID and an interface. Modifying other command parameters are optional.

Example:

The following example configures an area at the OSPF level:

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
    protocols {
        ospf {
            instance default {
                version ospf-v2
                area 1.1.1.1 {
                }
            }
        }
    }
```

9.1.4 Configuring a stub area

You can configure stub areas to control external advertisement flooding and to minimize the size of the topological databases on an area's routers. A stub area cannot also be configured as an NSSA. By default, summary route advertisements are sent into stub areas.

Example:

The following example configures an OSPF stub area:

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
    protocols {
        ospf {
            instance default {
                version ospf-v2
                area 1.1.1.1 {
                    stub {
                    }
                }
            }
        }
    }
```

9.1.5 Configuring a Not-So-Stubby area

You can explicitly configure an area to be a Not-So-Stubby Area (NSSA). NSSAs are similar to stub areas in that no external routes are imported into the area from other OSPF areas. An NSSA has the capability to flood external routes it learns throughout its area and by an area border router to the entire OSPF domain. An area cannot be both a stub area and an NSSA.

Example:

The following is an OSPF NSSA configuration example:

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
    protocols {
        ospf {
            instance default {
                version ospf-v2
                area 1.1.1.1 {
                    nssa {
                    }
                }
            }
        }
    }
```

9.1.6 Configuring an interface

You can configure an interface to act as a connection between a router and one of its attached networks. An interface includes state information that was obtained from underlying lower level protocols and from the routing protocol itself. An interface to a network is associated with a single IP address and mask (unless the network is an unnumbered point-to-point network). If the address is merely changed, then the OSPF configuration is preserved.

Example:

The following is an OSPF interface configuration example:

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
    protocols {
        ospf {
            instance default {
                version ospf-v2
                area 1.1.1.1 {
                    interface ethernet-1/2 {
                    }
                }
            }
        }
    }
```

10 BGP

Border Gateway Protocol (BGP) is an inter-AS routing protocol. An AS (autonomous system) is a network or a group of routers logically organized and controlled by common network administration. BGP enables routers to exchange network reachability information, including information about other ASs that traffic must traverse to reach other routers in other ASs.

ASs share routing information, such as routes to each destination and information about the route or AS path, with other ASs using BGP. Routing tables contain lists of known routers, reachable addresses, and associated path cost metrics for each router. BGP uses the information and path attributes to compile a network topology.

To set up BGP routing, participating routers must have BGP enabled, and be assigned to an AS, and the neighbor (peer) relationships must be specified. A router typically belongs to only one AS.

This section describes the minimal configuration necessary to set up BGP on SR Linux. This includes the following:

- Global BGP configuration, including specifying the autonomous system number (ASN) of the router, as well as the router ID.
- BGP peer group configuration, which specifies settings that are applied to BGP neighbor routers in the peer group.
- BGP neighbor configuration, which specifies the peer group to which each BGP neighbor belongs, as well as settings specific to the neighbor, including the AS to which the router is peered.

For information about all other BGP settings, see the SR Linux online help, as well as the *SR Linux Advanced Solutions Guide* and the *SR Linux Data Model Reference*.

10.1 BGP global configuration

Global BGP configuration includes specifying the autonomous system number (ASN) of the router and the router ID.

10.1.1 Configure an ASN

An autonomous system number (ASN) is a globally unique value that associates a router to a specific AS. Each router participating in BGP must have an ASN specified.

Example:

The following example configures an ASN for a router:

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                autonomous-system 65002
            }
        }
```

```
    }
```

10.1.2 Configure the router ID

The router ID, expressed like an IP address, uniquely identifies the router and indicates the origin of a packet for routing information exchanged between autonomous systems. The router ID is configured at the BGP level.

Example:

The following example configures a router ID:

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                router-id 2.2.2.2
        }
    }
```

10.2 Configuring a BGP peer group

A BGP peer group is a collection of related BGP neighbors. The group name should be a descriptive name for the group.

All parameters configured for a peer group are inherited by each peer (neighbor) in the peer group, but a group parameter can be overridden for specific neighbors in the configuration of that neighbor.

Example:

The following example configures the administrative state and trace options for a BGP peer group. These settings apply to all of the BGP neighbors that are members of this group, unless specifically overridden in the neighbor configuration.

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                group headquarters1 {
                    admin-state enable
                    traceoptions {
                        flag events {
                        }
                        flag graceful-restart {
                        }
                    }
                }
            }
```

10.3 Configuring BGP neighbors

After you configure a BGP group name and assign options, you can add neighbors within the same autonomous system to create internal BGP (iBGP) connections and/or neighbors in different autonomous systems to create external BGP (eBGP) peers. All parameters configured for the peer group to which the neighbor is assigned are applied to the neighbor, but a group parameter can be overridden on a specific neighbor basis.

Example:

The following example configures parameters for two BGP neighbors. The **peer-group** parameter configures both nodes to use the settings specified for the `headquarters1` group. The group settings apply unless they are specifically overridden in the neighbor configuration.

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                neighbor 192.168.11.1 {
                    peer-group headquarters1
                    description "default network-instance bgp neighbor to Node A"
                    peer-as 65001
                    local-as 65002 {
                    }
                    multihop {
                        admin-state enable
                        maximum-hops 3
                    }
                    failure-detection {
                        enable-bfd true
                        fast-failover true
                    }
                }
                neighbor 192.168.13.2 {
                    peer-group headquarters1
                    description "default network-instance bgp neighbor to Node C"
                    peer-as 65003
                    local-as 65002 {
                    }
                    failure-detection {
                        enable-bfd true
                        fast-failover true
                    }
                }
            }
        }
    }
```

10.4 eBGP multihop

External BGP (eBGP) multihop can be used to form adjacencies when eBGP neighbors are not directly connected to each other; for example, when a non-BGP router is between the eBGP neighbors.

BGP TCP/IP packets sent toward an eBGP neighbor by default have a TTL value of 1. If the BGP TCP/IP packets need to pass through more than one router to reach their destination, the TTL decrements to 0, and the packets are dropped.

To prevent this, you can enable multihop for the eBGP neighbor and specify the maximum number of hops for BGP TCP/IP packets sent to the neighbor. This allows the eBGP neighbor to be indirectly connected by up to the specified number of hops.

When multihop is not enabled, the IP TTL for eBGP sessions is set to 1, and the IP TTL for iBGP sessions is set to 64. By enabling multihop and configuring the maximum number of hops to a neighbor, it allows an eBGP session to have multiple hops, and an iBGP session to have a single hop, if required.

If multihop is enabled and the maximum-hops parameter is configured for a BGP peer group, the settings are applied to the members of the group. If the multihop configuration for a neighbor is changed, the session with the neighbor must be disconnected and re-established for the change to take effect.

10.4.1 Configuring eBGP multihop

To configure eBGP multihop, you enable it for the eBGP neighbor, and specify a value for the **maximum-hops** parameter. Additionally, the next-hop to the neighbor must be configured so that the two systems can establish a BGP session.

Example:

The following example enables multihop for an eBGP neighbor. The **maximum-hops** parameter is set to 2, which increases the TTL for BGP TCP/IP packets sent toward the eBGP neighbor, allowing the neighbor to be indirectly connected by up to 2 hops.

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                neighbor 192.168.11.1 {
                    multihop {
                        admin-state enable
                        maximum-hops 2
                    }
                }
        }
    }
```

Example

The following example configures a route to the next-hop toward the eBGP neighbor:

```
--{ * candidate shared default }--[  ]--
# info network-instance default static-routes
    network-instance default {
        static-routes {
            route 192.168.11.0/24 {
                next-hop-group static-ipv4-grp
            }
        }
    }
--{ * candidate shared default }--[  ]--
# info network-instance default next-hop-groups group static-ipv4-grp
    network-instance default {
        next-hop-groups {
            group static-ipv4-grp {
                nexthop 1 {
                    ip-address 192.168.22.22
                }
            }
```

```
        }
    }
```

10.5 AS path options

You can set the following options for handling the AS_PATH in received BGP routes:

- Allow own AS – configures the router to process received routes when its own ASN appears in the AS_PATH.
- Replace peer AS – configures the router to replace the ASN of the peer router in the AS_PATH with its own ASN.
- Remove private AS path numbers – configures the router to either delete private AS numbers, shortening the AS path length, or replace private AS numbers with the local AS number used toward the peer, maintaining the AS path length.

10.5.1 Configuring allow-own-as

Normally, when the ASN of a router appears in the AS_PATH of received routes, it is considered a loop, and the routes are discarded. The **allow-own-as** option configures the router to process the received routes when its own ASN appears in the AS_PATH. Specifically, it configures the maximum number of times the global ASN of the router can appear in any received AS_PATH before it is considered a loop and considered invalid. Default is 0.

Example:

The following example configures the router to process received routes where its own ASN appears in the AS_PATH a maximum of 1 time:

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                autonomous-system 65001
                as-path-options {
                    allow-own-as 1
                    }
                }
        }
    }
```

10.5.2 Configuring replace-peer-as

Normally, two sites having the same ASN would not be able to reach each other directly because the receiving router would see its own ASN in the AS_PATH and consider it a loop. To overcome this, you can configure the router to replace the peer ASN in the AS_PATH with its own ASN. When the **replace-peer-as** option is set to `true`, the router replaces every occurrence of the peer AS number that is present in the advertised AS_PATH with the local ASN used toward the peer.

Example:

The following example configures the router to replace the ASN of the peer with its own ASN:

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                as-path-options {
                    replace-peer-as true
                    }
                }
            }
        }
    }
```

10.5.3 Configuring remove-private-as

You can configure how the router handles private AS numbers: either delete them, shortening the AS path length, or replace private AS numbers with the local AS number used toward the peer, which maintains the AS path length.

You can configure the router to delete or replace private AS numbers that appear before the first occurrence of a non-private ASN in the sequence of most recent ASNs in the AS path. You can also configure the router to ignore private AS numbers when they are the same as the peer ASN.

Example:

The following example configures the router to delete private AS numbers (2-byte and 4-byte) from the advertised AS path toward all peers. This shortens the AS path.

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                as-path-options {
                    remove-private-as {
                        mode delete
                        }
                    }
                }
           }
        }
    }
```

Example

The following example configures the router to replace private AS numbers with the local AS number used toward the peer. This keeps the AS path the same length.

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                as-path-options {
                    remove-private-as {
                        mode replace
```

```
                        }
                    }
                }
        }
    }
```

Example

The following example configures the router to replace only private AS numbers that appear before the first occurrence of a non-private ASN in the sequence of most recent ASNs in the AS path.

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                as-path-options {
                    remove-private-as {
                        mode replace
                        leading-only true
                        }
                    }
                }
        }
    }
```

Example

The following example configures the router to ignore private AS numbers (neither delete nor replace them) when they are the same as the peer AS number.

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                as-path-options {
                    remove-private-as {
                        mode replace
                        ignore-peer-as true
                        }
                    }
                }
        }
    }
```

10.6 Route reflection

In a standard iBGP configuration, all BGP speakers within an AS must have full BGP mesh to ensure that all externally learned routes are redistributed through the entire AS.

Configuring route reflection provides an alternative to the full BGP mesh requirement: instead of peering with all other iBGP routers in the network, each iBGP router only peers with a router configured as a route reflector.

An AS can be divided into multiple clusters, with each cluster containing at least one route reflector, which redistributes routes to the clients in the cluster. The clients within the cluster do not need to maintain a full peering mesh between each other. They only require a peering to the route reflectors in their cluster. The route reflectors must maintain a full peering mesh between all non-clients within the AS.

10.6.1 Configuring route reflection

To configure a route reflector, you assign it a cluster ID and specify which neighbors are clients and which are non-clients. Clients receive reflected routes, and non-clients are treated as a standard iBGP peer.

Example:

The following example configures the router to be a route reflector for two clients SRL-1 and SRL-2. The router is assigned cluster ID 0.0.0.1.

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                route-reflector {
                    cluster-id 0.0.0.1
                    }
                }
                neighbor SRL-1 {
                    route-reflector {
                        cluster-id 0.0.0.1
                        client true
                    }
                neighbor SRL-2 {
                    route-reflector {
                        cluster-id 0.0.0.1
                        client true
                    }
                }
            }
        }
    }
```

10.7 Graceful restart

Graceful restart allows a router whose control plane has temporarily stopped functioning because of a system failure or a software upgrade to return to service with minimal disruption to the network.

To do this, the router relies on neighbor routers, which have also been configured for graceful restart, to maintain forwarding state while the router restarts. These neighbor routers are known as helper routers. The helper routers and the restarting router continue forwarding traffic using the previously learned routing information from the restarting router. Other routers in the network are not notified about the restarting router, so network traffic is not disrupted.

When graceful restart is enabled on the SR Linux and its neighbor, the two routers exchange information about graceful restart capability, including the Address Family Identifier (AFI) and Subsequent Address Family Identifier (SAFI) of the routes supported for graceful restart.

While the router restarts, the helper router marks the routes from the restarting router as stale, but continues to use them for traffic forwarding. When the BGP session is reestablished, the restarting router indicates to the helper router that it has restarted. The helper router then sends the restarting router any BGP RIB updates, followed by an End-of-RIB (EOR) marker indicating that the updates are complete. The restarting router then makes its own updates and sends them to the helper router, followed by an EOR marker.

Graceful restart is used in conjunction with the In-Service Software Upgrade (ISSU) feature, which can be used to upgrade 7220 IXR-D2 and D3 systems while maintaining non-stop forwarding. During the ISSU, a warm reboot brings down the control and management planes while the NOS reboots, and graceful restart maintains the forwarding state in peers. You can use a **tools** command to validate that the SR Linux and its peers support warm reboot, including graceful restart configuration. See the *SR Linux Software Installation Guide* for more information.

10.7.1 Configuring graceful restart

The following example enables graceful restart for the BGP instance. The SR Linux operates as a helper router for neighbor routers when they are restarting, assuming graceful restart is also enabled on the neighbors. Enabling graceful restart also indicates to the neighbors that they can serve as helper routers when the SR Linux itself is restarting.

Example:

When operating as a helper router, the SR Linux marks the routes from the restarting router as stale, but continues to use them for forwarding for a period of time while the neighbor router restarts. After this period expires, the SR Linux deletes the routes. The **stale-routes-time** parameter configures the amount of time in seconds the routes remain stale before they are deleted.

```
--{ * candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                graceful-restart {
                    admin-state enable
                    stale-routes-time 300
                }
            }
        }
    }
```

Following a restart, by default the system waits 600 seconds (10 minutes) to receive EOR markers from all helper routers for all address families that were up before the restart. After this time elapses, the system assumes convergence has occurred and sends its own EOR markers to its peers. You can configure the amount of time the system waits to receive EOR markers to be from 0 to 3,600 seconds.

For example, the following configures the amount of time the system waits to receive EOR markers to 270 seconds.

```
--{ * candidate shared default }--[  ]--
# info system warm-reboot
    system {
        warm-reboot {
            bgp-max-wait 270
        }
    }
```

10.8 BGP configuration management

Managing the BGP configuration on SR Linux can include the following tasks:

- Modifying an AS number

- Deleting a BGP neighbor from the configuration
- Deleting a BGP group
- Resetting BGP peer connections

10.8.1 Modifying an ASN

You can modify the ASN on the router, but the new ASN does not take effect until the BGP instance is restarted, either by administratively disabling/enabling the BGP instance, or by rebooting the system with the new configuration.

Example:

```
--{ * candidate shared default }--[ network-instance default ]--
# protocols bgp autonomous-system 95002
# protocols bgp admin-state disable
# protocols bgp admin-state enable
```

All established BGP sessions are taken down when the BGP instance is disabled.

10.8.2 Deleting a neighbor

Use the **delete** command to delete a BGP neighbor from the configuration.

Example:

```
--{ * candidate shared default }--[ network-instance default ]--
# delete protocols bgp neighbor 192.168.11.1
```

10.8.3 Deleting a group

Use the **delete** command to delete the settings for a BGP peer group from the configuration.

Example:

```
--{ * candidate shared default }--[ network-instance default ]--
# delete protocols bgp group headquarters1
```

10.8.4 Resetting BGP peer connections

To refresh the connections between BGP neighbors, you can issue a hard or soft reset. A hard-reset tears down the TCP connections and returns to IDLE state. A soft-reset sends route-refresh messages to each peer. The hard or soft reset can be issued to a specific peer, to peers in a specific peer-group, or to peers with a specific ASN.

Example:

The following command hard-resets the connections to the BGP neighbors in a peer group that have a specified ASN. The hard-reset applies both to configured peers and dynamic peers.

```
# tools network-instance default protocols bgp group headquarters1 reset-peer peer-as 95002
```

```
/network-instance[name=default]/protocols/bgp/group[group-name=headquarters1]:
    Successfully executed the tools clear command.
```

Example

The following command soft-resets the connection to BGP neighbors that have a specified ASN. The soft-reset applies both to configured peers and dynamic peers.

```
# tools network-instance default protocols bgp soft-clear peer-as 95002
/network-instance[name=default]/protocols/bgp:
    Successfully executed the tools clear command.
```

10.9 Protocol authentication

On the SR Linux, authentication of routing control messages for BGP, as well as other protocols such as LDP and IS-IS, is done using shared keys.

Message authentication between two routers involves sharing knowledge of a secret key and a cryptographic algorithm, such as MD5. This secret key, together with the message data, are used to generate a message digest. The message digest is added to each message transmitted by the sender and validated by the receiver, with the expectation that only a sender in possession of the secret key and algorithm details could generate the same message digest computed by the receiver of the message.

To limit exposure in the event a key is compromised, the secret key is changed at regular intervals using keys configured in a keychain. A keychain defines a list of one or more keys; each key is associated with a secret string, an algorithm identifier, and a start time.

When a protocol references a keychain for securing its messages with a set of peers, it uses the active key in the keychain with the most recent start time to generate the message digest in its sent messages, and it drops every received message that does not have an acceptable message digest.

10.9.1 Configuring protocol authentication

To configure protocol authentication, you configure an authentication keychain at the system level and configure a protocol to use the keychain. All protocol authentication is done using keychains. If a protocol requires authentication with a single neighbor using a single key, the key is configured within a keychain, and the protocol references the keychain.

Example:

The following example configures a keychain consisting of two keys.

```
--{ candidate shared default }--[  ]--
# info system authentication
    system {
        authentication {
            keychain k1 {
                tolerance 10 {
                key 1 {
                    admin-state enable
                    algorithm md5
                    authentication-key ZcvSElJzJx/wBZ9biCt
                    start-time 2020-05-26T10:21:01Z
                }
                key 2 {
                    admin-state enable
```

```
                    algorithm md5
                    authentication-key e7xdKlYO2DOm7v3IJv
                    start-time 2020-05-10T10:21:01Z
                }
            }
        }
```

Example

The following example configures BGP to use the keys in the keychain above for protocol authentication:

```
--{ candidate shared default }--[  ]--
# info network-instance default protocols bgp authentication
    network-instance default {
        protocols {
            bgp {
                authentication {
                    keychain k1
                    }
                }
            }
        }
    }
```

10.10 BGP shortcuts

With BGP shortcuts, SR Linux can include LDP LSPs or segment routing (SR-ISIS) tunnels in the BGP algorithm calculations. In this case, tunnels operate as logical interfaces directly connected to remote nodes in the network. Because the BGP algorithm treats the tunnels in the same way as a physical interface (being a potential output interface), the algorithm can select a destination node together with an output tunnel to resolve the next-hop, using the tunnel as a shortcut through the network to the destination.

Note: BGP shortcuts can only be used for next-hop resolution of IPv4-unicast RIB-Ins with an IPv4 next-hop address.

BGP next-hop resolution describes the procedures that BGP uses to resolve the next-hop address of each BGP RIB-In that forms part of a BGP route. The following table defines BGP RIB-In and BGP route in the context of BGP next-hop resolution.

Table 9: BGP RIB-IN and BGP route

BGP Term	Definition
BGP RIB-In	One of the following: • a received IPv4-unicast BGP route with an IPv4 next-hop address • a received IPv4-unicast BGP route with an IPv6 next-hop address (allowed as a result of sending an extended-nh-encoding capability to the peer) • a received IPv6-unicast BGP route with an IPv6 next-hop address
BGP route	A route submitted by BGP to the fib_mgr that resulted from the grouping of one or more BGP RIB-Ins. (Multiple BGP RIB-Ins per route describes a multipath scenario.)

With BGP shortcuts enabled, next-hop resolution determines whether to use a local interface or a tunnel to resolve the BGP next-hop.

Tunnel resolution mode

As part of the configuration for BGP shortcuts, you must define the tunnel-resolution mode (prefer/required/disabled). This mode determines the order of preference and fallback of using tunnels in the tunnel table to resolve the next-hop instead of using routes in the FIB, as described in the following sections.

Next-Hop Resolution of IPv4-Unicast RIB-Ins with IPv4 next-hop

The following table describes the next-hop resolution steps for IPv4-Unicast RIB-Ins with IPv4 next-hops, depending on the specified tunnel resolution mode.

Table 10: Next-hop resolution for IPv4 Unicast RIB-Ins with IPv4 next-hop address

Tunnel Resolution Mode	Next-hop resolution steps in BGP
prefer	**1.** Start with TTM lookup: **a.** Find all the tunnels in TTM with an endpoint that matches the BGP next-hop address and that have one of the types listed in the allow list. **b.** If there is a single tunnel, select that tunnel. The RIB-IN is resolved; exit. **c.** If there are multiple tunnels, select the tunnel with the numerically lowest TTM preference, and if a further tie-break is needed, select the tunnel with the lowest TTM metric. The RIB-IN is resolved; exit. **2.** If there are no tunnels, fallback to FIB lookup: **a.** Find the longest match active route in the FIB that matches the BGP next-hop address. There are presently no restrictions on this route; it can be an IGP route, a static blackhole route, a default route, or another BGP route. **b.** If there is a longest match route and it eventually resolves to a blackhole next-hop, interface or tunnel then the RIB-IN is resolved; exit. **c.** If there is no matching route the RIB-IN is unresolved.
require	Perform TTM lookup only, as described in 1 above. If there is no matching tunnel, the RIB-IN is unresolved.
disabled	Perform FIB lookup only, as described in 2 above.

Next-Hop Resolution of IPv4-Unicast and IPv6-Unicast RIB-Ins with IPv6 next-hop

If the next-hop address for the IPv4-Unicast RIB-In is an IPv6 address, the next-hop is resolved by the longest prefix match IPv6 route in the FIB. This is the only option because there are no IPv6 tunnels in the TTM. The same logic applies to BGP RIB-Ins with IPv6-unicast NLRI address family as they can only have an IPv6 next-hop address. The next-hop resolution logic is the same as the FIB lookup described in the preceding table.

10.10.1 Configuring BGP shortcuts

To configure BGP shortcuts, you must configure the BGP protocol in the default network-instance with the allowed tunnel types for next-hop resolution. You must also define the tunnel resolution mode, which determines the order of preference and the fallback when using tunnels in the tunnel table instead of routes in the FIB. Available options are as follows:

- require: requires tunnel table lookup instead of FIB lookup
- prefer: prefers tunnel table lookup over FIB lookup
- disabled (default): performs FIB lookup only

Example: Configure BGP next-hop resolution to allow segment routing tunnels

The following example shows the BGP next-hop resolution configuration to allow SR-ISIS tunnels, with the tunnel mode set to `prefer`.

```
--{ * candidate shared default }--[ ]--
# info network-instance default protocols bgp ipv4-unicast next-hop-resolution ipv4-next-hops
 tunnel-resolution
    network-instance default {
        protocols {
            bgp {
                ipv4-unicast {
                    next-hop-resolution {
                        ipv4-next-hops {
                            tunnel-resolution {
                                mode prefer
                                allowed-tunnel-types [
                                    sr-isis
                                ]
                            }
                        }
                    }
                }
            }
        }
    }
```

11 IS-IS

Intermediate System to Intermediate System (IS-IS) is a link-state IGP that uses the Shortest Path First (SPF) algorithm to determine routes. Routing decisions are made using the link-state information. IS-IS evaluates topology changes and, if necessary, performs SPF recalculations.

Entities within IS-IS include networks, intermediate systems, and end systems. In IS-IS, a network is an Autonomous System (AS), or routing domain, with end systems and intermediate systems. A router is an intermediate system. End systems are network devices that send and receive protocol data units (PDUs). Intermediate systems send, receive, and forward PDUs.

End system and intermediate system protocols allow routers and nodes to identify each other. IS-IS sends out link-state updates periodically throughout the network, so each router can maintain current network topology information.

IS-IS supports large ASs by using a two-level hierarchy. A large AS can be administratively divided into smaller, more manageable areas. A system logically belongs to one area. Level 1 routing is performed within an area. Level 2 routing is performed between areas. Routers can be configured as Level 1, Level 2, or both Level 1/2.

On SR Linux, you can configure a single named IS-IS instance per network instance. The following summarizes SR Linux support for IS-IS:

- Level 1, Level 2, and Level 1/2 IS types
- Configurable network entity title (NET) per IS-IS instance.
- Support for IPv4/v6 routing
- ECMP with up to 128 next hops per destination
- IS-IS export policies (redistribution of other types of routes into IS-IS)
- Authentication of CSNP, PSNP, and IIH PDUs with authentication type and key, configurable per instance and per instance level. Authentication type and key for IIH PDUs also configurable per interface and level.
- Support for authentication keychains
- Purge Originator ID TLV (RFC 6232)
- Options to ignore and suppress the attached bit
- Ability to set the overload bit immediately or to set the bit after each subsequent restart of the IS-IS manager application and leave it on for a configurable duration each time
- Control over the Link-state PDU (LSP) MTU size, with range from 490 bytes to 9490 bytes
- Configuration control over timers related to LSP lifetime, LSP refresh interval, SPF calculation triggers, and LSP generation
- Support for hello padding (strict, loose and adaptive modes)
- Support for graceful restart, but only acting as a helper of the restarting router
- Level 1 to Level 2 route summarization
- BFD for fast failure detection
- Configurable hello timer/multiple per interface and level
- Support for wide metrics (configurable per level)

- Configurable route preference for each route type, Level 1-internal, Level 1-external, Level 2-internal and Level 2-external.

The **info detail** command displays default values for an IS-IS instance on SR Linux. For example:

```
--{ * candidate shared default }--[ network-instance default protocols isis ]--
# info detail
    instance i1 {
        admin-state disable
        level-capability L2
        max-ecmp-paths 1
        poi-tlv false
        attached-bit {
            ignore false
            suppress false
        }
        overload {
            advertise-interlevel false
            advertise-external false
            immediate {
                set-bit false
                max-metric false
            }
            on-boot {
                set-bit false
                max-metric false
            }
        }
        timers {
            lsp-lifetime 1200
            lsp-refresh {
                interval 600
                half-lifetime true
            }
            spf {
                initial-wait 1000
                second-wait 1000
                max-wait 10000
            }
            lsp-generation {
                initial-wait 1000
                second-wait 1000
                max-wait 5000
            }
        }
        transport {
            lsp-mtu-size 1492
        }
        ipv4-unicast {
            admin-state enable
        }
        ipv6-unicast {
            admin-state enable
            multi-topology false
        }
        graceful-restart {
            helper-mode false
        }
        auto-cost {
        }
        authentication {
            csnp-authentication false
            psnp-authentication false
            hello-authentication false
        }
```

```
        inter-level-propagation-policies {
            level1-to-level2 {
            }
        }
    }
```

11.1 Basic IS-IS configuration

To configure IS-IS, perform the following tasks:

- Enable an IS-IS instance
- If necessary, modify the level capability on the global IS-IS instance level
- Define area addresses
- Configure IS-IS interfaces

11.1.1 Enabling an IS-IS instance

SR Linux supports a single IS-IS instance within a network instance. The following example enables an IS-IS instance within the default network instance.

Example:

```
--{ * candidate shared default }--[ network-instance default protocols ]--
# info isis
 isis {
        instance i1 {
        }
    }
```

11.1.2 Configuring the router level

When IS-IS is enabled, the default level-capability value is Level 1/2. This means that the router operates with both Level 1 and Level 2 routing capabilities. To change the default value in order for the router to operate as a Level 1 router or a Level 2 router, you must explicitly modify the level value.

The level-capability value can be configured on the global IS-IS instance level and also on the interface level. The level-capability value determines which level values can be assigned on the router level or on an interface-basis.

In order for the router to operate as a Level 1 only router or as a Level 2 only router, you must explicitly specify the level-number value.

- Specify Level 1 to route only within an area
- Specify Level 2 to route to destinations outside an area, toward other eligible Level 2 routers

Example:

The following example configures the level capability for an IS-IS instance to Level 2.

```
--{ * candidate shared default }--[ network-instance default protocols ]--
# info isis
 isis {
```

```
        instance i1 {
            level-capability L2
        }
    }
```

11.1.3 Configuring the Network Entity Title

SR Linux supports a configurable network entity title (NET) per IS-IS instance. The NET is 8-20 octets long and consists of 3 parts: the area address (1-13 octets), the system ID (6 octets), and the n-selector (1 octet, must be 00)

The area address portion of the NET defines the IS-IS area to which the router belongs. At least one area address should be configured on each router participating in IS-IS.

The area address portion of the NET identifies a point of connection to the network, such as a router interface. The routers in an area manage routing tables about destinations within the area. The NET value is used to identify the IS-IS area to which the router belongs.

The NET value is divided into three parts. Only the Area ID portion is configurable.

1. Area ID — A variable length field between 1 and 13 bytes. This includes the Authority and Format Identifier (AFI) as the most significant byte and the area ID.

2. System ID — A 6-byte system identifier. This value is not configurable. The system ID is derived from the system or router ID.

3. Selector ID — A 1-byte selector identifier that must contain zeros when configuring a NET. This value is not configurable. The selector ID is always 00.

Example:

The following example configures a NET for an IS-IS instance:

```
--{ * candidate shared default }--[ network-instance default protocols ]--
# info isis
 isis {
        instance i1 {
            net 49.0001.1921.6800.1002.00
            }
        }
    }
```

11.1.4 Configuring global parameters

Commands and parameters configured on the global IS-IS instance level are inherited by the interface levels. Parameters specified in the interface and interface-level configurations take precedence over global configurations.

Example:

The following example shows the command usage to configure global-level IS-IS. The authentication setting references a keychain defined at the system level (see Protocol authentication).

```
--{ * candidate shared default }--[ network-instance default protocols ]--
# info isis
    isis {
        instance i1 {
            level-capability L2
```

```
            overload {
                on-boot {
                    timeout 90
                }
            }
            authentication {
                keychain isisglobal
            }
        }
    }
```

11.1.5 Configuring interface parameters

There are no interfaces associated with IS-IS by default. An interface belongs to all areas configured on a router. Interfaces cannot belong to separate areas. There are no default interfaces applied to the router IS-IS instance. You must configure at least one IS-IS interface in order for IS-IS to work.

You can configure both the Level 1 parameters and the Level 2 parameters on an interface. The level-capability value determines which level values are used.

Example:

The following example configures interface parameters for an IS-IS instance:

```
--{ * candidate shared default }--[ network-instance default protocols isis ]--
# info instance i1
    instance i1 {
        interface ethernet-1/2.1 {
            circuit-type point-to-point
            ipv4-unicast {
                admin-state enable
            }
            level 1 {
                authentication {
                    hello-authentication true
                }
            }
        }
        level 1 {
        }
    }
```

11.2 Displaying IS-IS information

Use the following **show** commands to display the following information for an IS-IS instance running in a specified network instance.

- Interface information
- Adjacency information
- IS-IS link state database information

Example:

To display summary information for an IS-IS instance:

```
# show network-instance default protocols isis interface
```

```
-------------------------------------------------------------------------------------------------
---
Network-instance    : default
IS-IS instance      : global
-------------------------------------------------------------------------------------------------
---
Interface      Oper-State Level   Circuit-Id Circuit-Type  L1/L2 Metric
lo0.0          up         L2      1          p2p           -/0
ethernet-1/1.0 up         L1L2    2          p2p           10/100
-------------------------------------------------------------------------------------------------
---
Interfaces: 2
-------------------------------------------------------------------------------------------------
---
```

Example

To display details for each interface:

```
# show network-instance default protocols isis interface detail
-------------------------------------------------------------------------------------------------
---
Network-instance    : default
IS-IS instance      : global
-------------------------------------------------------------------------------------------------
---
Interface: lo0.0
  Status          : IS-IS is admin enabled, oper up
  Circuit id/type : id 1, passive
  Hello auth      : disabled
  Hello padding   : disabled
  CSNP interval   : n/a
  LSP pacing      : n/a
  -----------------------------------------------------------------------------------------------
---
  Level 1 is disabled
  -----------------------------------------------------------------------------------------------
---
  Level 2 is enabled
  Adjacencies         : 0
  Designated IS       : n/a
  Hello auth          : disabled
  DIS priority        : 64
  Hello interval      : 9
  Hello multiplier    : 3
  Metric              : 0
  IPv6-unicast metric : 0
  -----------------------------------------------------------------------------------------------
---
-------------------------------------------------------------------------------------------------
---
Interface: ethernet-1/1.0
  Status          : IS-IS is admin enabled, oper up
  Circuit id/type : id 2, point-to-point
  Hello auth      : enabled using keychain "test"
  Hello padding   : strict
  CSNP interval   : 10 seconds
  LSP pacing      : 100 ms
  -----------------------------------------------------------------------------------------------
---
  Level 1 is enabled
  Adjacencies         : 1
  Designated IS       : n/a
  Hello auth          : enabled using keychain "test"
  DIS priority        : 64
```

```
  Hello interval      : 9
  Hello multiplier    : 3
  Metric              : 10
  IPv6-unicast metric : 0
  -------------------------------------------------------------------------------------------
---
  Level 2 is enabled
  Adjacencies         : 1
  Designated IS       : n/a
  Hello auth          : enabled using keychain "test"
  DIS priority        : 64
  Hello interval      : 9
  Hello multiplier    : 3
  Metric              : 100
  IPv6-unicast metric : 0
  -------------------------------------------------------------------------------------------
---
-------------------------------------------------------------------------------------------
---
Interfaces: 2
-------------------------------------------------------------------------------------------
---
```

Example

To display IS-IS adjacency information:

```
# show network-instance default protocols isis adjacency
-------------------------------------------------------------------------------------------
---
Network-instance    : default
IS-IS instance      : global
-------------------------------------------------------------------------------------------
---
System-Id      Adj-Level  Interface        IPv4-Address  State  Uptime       Rem-Hold
<hostname1>    L1         ethernet-1/1.0   10.0.0.1      Up     0d 00:46:43  19s
<hostname1>    L2         ethernet-1/1.0   10.0.0.1      Up     0d 00:46:43  19s
-------------------------------------------------------------------------------------------
---
Adjacencies: 2
-------------------------------------------------------------------------------------------
---
```

Example

To display information from the IS-IS link state database:

```
# show network-instance default protocols isis database
-------------------------------------------------------------------------------------------
---
Network-instance    : default
IS-IS instance      : global
-------------------------------------------------------------------------------------------
---
IS-IS level 1 link state database
LSP count: 3
-------------------------------------------------------------------------------------------
---
LSP ID       Sequence   Checksum   Lifetime   Attributes
R1.00-00     0x12       0x58b6     708        L1 L2 ATT OL
R2.00-00     0x140x39a6     834        L1 L2
R2.03-00     0x10       0xdee2     915        L1 L2
-------------------------------------------------------------------------------------------
---
```

```
IS-IS level 2 link state database
LSP count: 3
-------------------------------------------------------------------------------------------------
---
LSP ID       Sequence   Checksum   Lifetime   Attributes
R1.00-00     0x49       0x8850     810        L1 L2 OL
R2.00-00     0x490x9d04     906        L1 L2
R2.03-00     0x10       0xdbba     937        L1 L2
-------------------------------------------------------------------------------------------------
---
```

11.3 Clearing IS-IS information

To clear information for an IS-IS instance, use the **tools** commands below:

Example:

To clear statistics for an IS-IS instance running in a specified network instance:

```
# tools network-instance default protocols isis instance i1 statistics clear
```

Example

To clear link state database information for a level:

```
# tools network-instance default protocols isis instance i1 link-state-database clear
```

Example

To clear IS-IS adjacency information for an interface:

```
# tools network-instance default protocols isis instance i1 interface ethernet 1/2 adjacencies
 clear
```

12 BFD

Bidirectional Forwarding Detection (BFD) is a lightweight mechanism used to monitor the liveliness of a remote neighbor. It is meant to be lightweight enough so that the ongoing sending and receiving mechanism can be implemented in the forwarding hardware. Because of this lightweight nature, BFD can send and receive messages at a much higher rate than other control plane hello mechanisms. This attribute allows connection failures to be detected faster than other hello mechanisms.

SR Linux supports BFD asynchronous mode, where BFD control packets are sent between two systems to activate and maintain BFD neighbor sessions between them.

BFD can be configured to monitor connectivity for the following:

- BGP peers. See Configuring BFD under the BGP protocol.
- Next-hops for static routes. See Configuring failure detection for static routes.
- OSPF adjacencies. See Configuring BFD under OSPF.
- IS-IS adjacencies. See Configuring BFD under IS-IS.

SR Linux supports one BFD session per port/connector, or up to 1152 sessions for an 8-slot chassis, depending on the hardware configuration.

On SR Linux systems that support link aggregation groups (LAGs), SR Linux supports micro-BFD, where BFD sessions are established for individual members of a LAG. If the BFD session for one of the links indicates a connection failure, the link is taken out of service from the perspective of the LAG. See Micro-BFD.

This section describes the minimal configuration necessary to set up BFD on SR Linux. To create a BFD session, you configure BFD on both systems (or BFD peers). When BFD has been enabled on the interfaces and at the global level for the appropriate routing protocols, a BFD session is created, BFD timers are negotiated, and the BFD peers begin to send BFD control packets to each other at the negotiated interval.

12.1 Configuring BFD for a subinterface

You can enable BFD with an associated subinterface, and set values for intervals and criteria for declaring a session down.

Timer values are in microseconds. The detection interval for the BFD session is calculated by multiplying the value of the negotiated transmission interval by the value specified in this field.

Example:

The following example configures BFD for a subinterface.

```
--{ candidate shared default }--[  ]--
# info bfd
    bfd {
        subinterface ethernet-1/2.1 {
            admin-state enable
            desired-minimum-transmit-interval 250000
            required-minimum-receive 250000
            detection-multiplier 3
```

```
        }
    }
```

12.2 Configuring BFD under the BGP protocol

BFD can be configured under the BGP protocol at the global, group, or neighbor level.

Before it is enabled, BFD must first be configured for a subinterface and timer values must be set. See Configuring BFD for a subinterface.

Example:

The following example configures BFD under the BGP protocol at the global level.

```
--{ candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                failure-detection {
                    enable-bfd true
                    }
                }
            }
        }
    }
```

Example

The following example configures BFD for the links between peers within an associated BGP peer group.

```
--{ candidate shared default }--[  ]--
# info network-instance default protocols bgp
    network-instance default {
        protocols {
            bgp {
                group test {
                    failure-detection {
                        enable-bfd true
                        }
                    }
                }
            }
    }
```

Example

The following example configures BFD for the link between BGP neighbors.

```
--{ candidate shared default }--[  ]--
# info network-instance default protocols bgp
    network-instance default {
        protocols {
            bgp {
                neighbor 192.35.1.3 {
                    failure-detection {
                        enable-bfd true
                        fast-failover true
                    }
                }
```

```
            }
        }
    }
```

12.3 Configuring BFD under OSPF

For OSPF and OSPFv2, BFD can be enabled at the interface level to monitor the connectivity between the router and its attached network.

Example:

```
--{ candidate shared default }--[  ]--
# info network-instance default protocols ospf
    network-instance default {
        protocols {
            ospf {
                instance o1 {
                    area 1.1.1.1 {
                        interface ethernet-1/1.1 {
                            failure-detection {
                                enable-bfd true
                            }
                        }
                    }
                }
            }
        }
    }
```

12.4 Configuring BFD under IS-IS

BFD can be configured at the interface level for IS-IS. You can optionally configure a BFD-enabled TLV to be included for IPv4 or IPv6 on the IS-IS interface.

Example:

```
--{ candidate shared default }--[  ]--
# info network-instance default protocols isis
    network-instance default {
        protocols {
            isis {
                instance i1 {
                    ipv4-unicast {
                        admin-state enable
                    }
                    interface ethernet-1/1.1 {
                        ipv4-unicast {
                            enable-bfd true
                            include-bfd-tlv true
                        }
                    }
                }
            }
        }
    }
```

12.5 Viewing the BFD state

Use the **info from state** command to verify the BFD state for a network-instance.

Example:

```
# info from state bfd network-instance default peer 30
    bfd {
        network-instance default {
            peer 30 {
                oper-state up
                local-address 192.35.1.5
                remote-address 192.35.1.3
                remote-discriminator 25
                subscribed-protocols bgp_mgr
                session-state UP
                remote-session-state UP
                last-state-transition 2020-01-24T16:22:55.224Z
                failure-transitions 0
                local-diagnostic-code NO_DIAGNOSTIC
                remote-diagnostic-code NO_DIAGNOSTIC
                remote-minimum-receive-interval 1000000
                remote-control-plane-independent false
                active-transmit-interval 250000
                active-receive-interval 250000
                remote-multiplier 3
                async {
                    last-packet-transmitted 2020-01-24T16:23:19.385Z
                    last-packet-received 2020-01-24T16:23:18.906Z
                    transmitted-packets 32
                    received-packets 32
                    up-transitions 1
                }
            }
        }
    }
```

12.6 Micro-BFD

Micro-BFD refers to running BFD over the individual links in a link aggregation group (LAG) to monitor the bidirectional liveliness of the Ethernet links that make up the LAG.

A LAG member cannot be made operational within the LAG until the micro-BFD session is fully established. If a micro-BFD session fails, the corresponding Ethernet link is taken out of service from the perspective of the LAG.

Micro-BFD is supported on Ethernet LAG interfaces with an IP interface. Micro-BFD sessions are associated with each individual link. When enabled, the state of the individual links depends on the micro-BFD session state:

- Micro-BFD sessions must be established between both endpoints of a link before the link can be operationally up.
- If the micro-BFD session fails, the associated Ethernet link becomes operationally down from the perspective of the LAG.
- If LACP is not enabled for the LAG, if the Ethernet port is up, the system attempts to re-establish the micro-BFD session with the far end of the link.

- If LACP not enabled for the LAG, if the Ethernet port is up, the system attempts to re-establish the micro-BFD session with the far end of the link when LACP reaches distributing state.

If a link is not active for forwarding from the perspective of a LAG, ARP can still be performed across the link. For example, when a link is being brought up, and its micro-BFD session is not yet established, ARP can still be performed for the MAC address at the far end of the link, even though the link is not yet part of the LAG.

Micro-BFD packets bypass ingress and egress subinterface/interface ACLs, but received micro-BFD packets can be matched by CPM filters for filtering and logging.

Micro-BFD is supported on all SR Linux systems that also support LAGs: 7250 IXR; 7220 IXR-D1, D2, and D3; 7220 IXR-H2 and H3.

12.6.1 Configuring micro-BFD for a LAG interface

The following example configures micro-BFD for a LAG interface. The example configures IP addresses to be used as the source address for IP packets and a remote address for the far end of the BFD session.

The example configures the minimum interval in microseconds between transmission of BFD control packets, as well as the minimum acceptable interval between received BFD control packets. The detection-multiplier setting specifies the number of packets that must be missed to declare the BFD session as down.

Example:

```
--{ * candidate shared default }--[  ]--
# info bfd micro-bfd-sessions
    micro-bfd-sessions {
        lag-interface lag1 {
            admin-state enable
            local-address 192.35.2.5
            remote-address 192.35.2.3
            desired-minimum-transmit-interval 250000
            required-minimum-receive 250000
            detection-multiplier 3
        }
    }
```

12.6.2 Viewing the micro-BFD state

Use the **info from state** command to verify the micro-BFD state for members of a LAG interface.

Example:

```
# info from state micro-bfd-sessions lag-interface lag1 member-interface ethernet 2/1
    micro-bfd-sessions
        lag-interface lag1 {
            admin-state UP
            local-address 192.35.1.5
            remote-address 192.35.1.3
            desired-minimum-transmit-interval 250000
            required-minimum-receive 250000
            detection-multiplier 3
            member-interface ethernet 2/1 {
                session-state UP
                remote-session-state UP
                last-state-transition 2020-01-24T16:22:55.224Z
                last-failure-time 2020-01-24T16:22:55.224Z
```

```
                failure-transitions 0
                local-discriminator 25
                remote-discriminator 25
                local-diagnostic-code NO_DIAGNOSTIC
                remote-diagnostic-code NO_DIAGNOSTIC
                remote-minimum-receive-interval 1000000
                remote-control-plane-independent false
                active-transmit-interval 250000
                active-receive-interval 250000
                remote-multiplier 3
                async {
                   last-clear 2020-01-23T16:21:19.385Z
                   last-packet-transmitted 2020-01-24T16:23:19.385Z
                   last-packet-received 2020-01-24T16:23:18.906Z
                   transmitted-packets 32
                   received-errored-packets 3
                   received-packets 32
                   up-transitions 1
                }
             }
          }
       }
```

13 Routing policies

The SR Linux supports policy-based routing. Policy-based routing controls the size and content of the routing tables, the routes that are advertised, and the best route to take to reach a destination.

Each routing policy has a sequence of rules (called entries or statements) and a default action. Each statement has numerical sequence identifier that determines its order relative to other statements in that policy. When a route is analyzed by a policy, it is evaluated by each statement in sequential order.

Each policy statement has zero or more match conditions and a base action (either accept or reject); the statement may also have route-modifying actions. A route matches a statement if it meets all of the specified match conditions.

The first statement that matches the route determines the actions that are applied to the route. If the route is not matched by any statements, the default action of the policy is applied. If there is no default action, then a protocol- and context-specific default action is applied.

All routes match a statement with no match conditions. When a route fulfills the match conditions of a statement, the base action of the statement is applied, along with all of its route-modifying actions.

13.1 Creating a routing policy

A routing policy consists of one or more statements that contain the following:

- Match conditions, such as protocol type, AS path length, and address family, against which routes are evaluated
- Actions, such as accept or reject, for routes that match the conditions in the statement

Each statement has a specified sequence number. If a policy has multiple statements, they are processed in sequence-number order. In addition, you can specify a default action that applies to routes that do not match any statement in the policy.

13.1.1 Specifying match conditions

You can specify the following match conditions in a policy statement:

- Match routes by their protocol type – BGP, static, direct, host, IS-IS, OSPF, and so on.
- Match routes of any protocol by their address family – IPv4-unicast, IPv6-unicast, EVPN.
- Match routes of any protocol by their IPv4/IPv6 prefix.
- Match aggregate and BGP routes by their standard and large communities.
- Match BGP routes by their AS path length.
- Match BGP routes by their AS path encoding.
- Match BGP routes by whether the EVPN route has an Ethernet Segment identifier (ESI) that matches a member of a specified ESI set.
- Match BGP routes by whether the EVPN route has an Ethernet tag ID that matches a member of a specified Ethernet tag set.

- Match BGP routes by whether the EVPN type-2 route has a MAC address that matches a member of a specified MAC set.
- Match BGP routes by whether the originating router's IP address in an EVPN type-3 or type-4 route matches a specified IP address.
- Match BGP routes by whether the EVPN route is a specified route type (ethernet-ad, mac-ip, ethernet-segment, imet, or ip-prefix).
- Match BGP routes by whether the BGP VPN route has a specified route distinguisher value.
- Match IS-IS routes by their association with either Level1 (L1) or Level2 (L2).
- Match IS-IS routes by their route type (internal or external). An IPv4 route is internal if the prefix was signaled in TLV 128. An IPv4 route is external if the prefix was signaled in TLV 130. The encoding of TLV 236 indicates whether an IPv6 route is internal or external.
- Match IS-IS routes by their tag value.
- Match OSPF routes by their area ID.
- Match OSPF routes by their route type (intra-area, inter-area, type-1-ext, type-2-ext, type-1-nssa, type-2-nssa, summary-aggregate, or nssa-aggregate).
- Match OSPF routes by instance ID.
- Match routes based on EVPN route type.
- Match routes based on IP addresses and prefixes via prefix-sets for route types 2 and 5.
- Match routes based BGP encapsulation extended community.

If a statement has no match conditions defined, all routes evaluated by the policy are considered to be matches.

Example:

The following example specifies BGP protocol as a match condition in a policy statement:

```
--{ candidate shared default }--[  ]--
# info routing-policy policy policy01
    routing-policy {
        policy policy01 {
            statement 100 {
                match {
                    protocol bgp
                }
            }
        }
    }
```

13.1.1.1 Specifying a list as a match condition

You can specify a list of items, such as address prefixes, autonomous systems, BGP communities, and Ethernet tags as match criteria. To do this, you configure the list and include the list in a policy statement.

Example:

The following example specifies a list of prefixes that can be used as a match condition in a policy statement:

```
--{ candidate shared default }--[  ]--
# info routing-policy
```

```
    routing-policy {
        prefix-set western {
            prefix 10.10.0.1/32 mask-length-range exact {
            }
            prefix 10.10.0.2/32 mask-length-range exact {
            }
            prefix 10.10.0.3/32 mask-length-range exact {
            }
            prefix 10.10.0.4/32 mask-length-range exact {
            }
        }
    }
```

13.1.2 Specifying actions

The following are valid actions for routes that meet the match conditions in a statement:

- Accept the route.
- Reject the route.
- Modify the AS path of a BGP route by prepending additional AS numbers.
- Modify the AS path of a BGP route by deleting the current AS path and replacing it with a new AS path (a sequence of zero or more AS numbers).
- Add, delete, or replace standard and large communities attached to a BGP route.
- Modify the LOCAL_PREF attribute of a BGP route by specifying a new value.
- Modify the ORIGIN attribute of a BGP route by specifying a new value.
- Set the next-hop-self option for a BGP route.
- Set the next-hop-unchanged option for a BGP route.
- Add, delete, or replace the route tag associated with an IS-IS route.
- Proceed to the next statement in the policy
- Proceed to the next policy

Example:

The following example specifies a policy with two statements. If a route matches the first statement, the action is to proceed to the next statement. If the route matches the second statement, the action is to accept.

```
--{ candidate shared default }--[  ]--
# info routing-policy
    routing-policy {
        policy policy02 {
            statement 100 {
                match {
                    protocol bgp
                }
                action {
                    next-entry
                }
            }
            statement 101 {
                match {
                    prefix-set western
                }
                action {
```

```
                    accept
                }
            }
        }
    }
```

13.1.3 Specifying a default action

You can optionally specify the policy action for routes that do not match the conditions defined in the policy. The default action can be set to all available action states including accept, reject, next-entry, and next-policy.

- If a default action is defined, and no matches occur with the entries in the policy, the default action is used.
- If no default action is specified, the default behavior of the protocol controls whether the routes match.

  For BGP, the default import action is to accept all routes from BGP. For internal routes, the default export action is to advertise them to BGP peers. For external routes, the default export action is not to advertise them to BGP peers.

Example:

The following example defines a policy where the default action for non-matching routes is to reject them:

```
--{ candidate shared default }--[  ]--
# info routing-policy
    routing-policy {
        policy policy01 {
            default-action {
                reject {
                }
            }
        }
    }
```

13.2 Applying a routing policy

Routing policies can be applied to routes received from other routers (imported routes), as well as routes advertised to other routers (exported routes). Routing policies can be applied at the network-instance level, peer-group level, and neighbor level.

Example:

The following example specifies that BGP in the default network-instance applies `policy01` to imported routes:

```
--{ candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                import-policy policy01
                }
            }
```

Example

The following example applies `policy02` to BGP routes exported from the peers in peer-group `headquarters1`:

```
--{ candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                group headquarters1 {
                    export-policy policy02
                    }
                }
            }
        }
```

Example

The following example applies `policy02` to BGP routes exported from a specific BGP neighbor:

```
--{ candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                neighbor 192.168.11.1 {
                    export-policy policy02
                }
        }
    }
```

13.2.1 Applying a default policy to eBGP sessions

You can specify the action to take for routes exported to or imported from eBGP peers to which no configured policy applies. This is set with the **ebgp-default-policy** command and the **export-reject-all** and **import-reject-all** parameters.

- The **export-reject-all** parameter, when set to **true**, causes all outbound routes that do not match a configured export policy to be rejected as if they had been rejected by a configured export policy. The default is **true**.
- The **import-reject-all** parameter, when set to **true**, causes all inbound routes that do not match a configured import policy to be rejected as if they had been rejected by a configured import policy. The default is **true**.

Example:

The following example allows a BGP neighbor to export a default route even though the route is not subject to any configured policy:

```
--{ candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                ebgp-default-policy {
                    export-reject-all false
                    }
                neighbor 2001:db8::c11 {
```

```
                    send-default-route {
                        ipv6-unicast true
                    }
                }
            }
        }
```

13.2.2 Replacing an AS path

You can configure a routing policy where the AS path in matching routes is replaced by a list of AS numbers specified in the policy. For routes that match the policy, the current AS path is deleted and replaced with an AS_SEQ element containing the AS numbers listed in the policy in their configured sequence.

If you configure an empty AS list in the policy, then the current AS path in a matching route is deleted, and it would then have a null AS_PATH attribute.

Example:

The following is an example of a routing policy whose action is to replace the AS path in matching routes.

```
--{ candidate shared default }--[  ]--
# info routing-policy
    routing-policy {
        policy policy02 {
            statement 100 {
                match {
                    protocol bgp
                }
                action {
                    bgp {
                        as-path {
                            replace {
                            }
                        }
                    }
                }
            }
        }
    }
```

14 Access control lists

An Access Control List, or ACL, is an ordered set of rules that are evaluated on a packet-by-packet basis to determine whether access should be provided to a specific resource. ACLs can be used to drop unauthorized or suspicious packets from entering or leaving a routing device via specified interfaces.

SR Linux supports the following types of ACLs:

- Interface filters

  An interface filter is an IPv4 or IPv6 ACL that is applied to a routed or bridged subinterface to restrict traffic allowed to enter or exit the subinterface. An interface ACL can be applied to input or output traffic for one or more subinterfaces.

  See Interface filters for more information.

- Packet capture filters

  A packet capture filter is an IPv4 or IPv6 ACL used to extract packets to the control plane for inspection by packet capture tools. When an ingress IP packet on any line card transits through the router, and it matches a rule in a packet capture filter policy, it is copied and extracted toward the CPM and delivered to a Linux virtual Ethernet interface so that it can be displayed by a packet capture utility, or encapsulated and forwarded to a remote monitoring system.

  See Packet capture filters for more information.

- CPM filters

  A CPM filter is an IPv4 or IPv6 ACL used for control plane protection. There is one CPM filter for IPv4 traffic and another CPM filter for IPv6 traffic. When an ingress IP packet is matched by a CPM filter rule, and it is a terminating packet (that is, it must be extracted to the CPM), then it is processed according to the matching CPM filter rule.

  See Control plane module (CPM) filters for more information.

- System filters (7220 IXR-D1, D2, and D3 systems only)

  A system filter ACL is an IPv4 or IPv6 ACL that is evaluated early in the ingress pipeline, at a stage before tunnel termination occurs and before interface filters are run. For VXLAN traffic, system filters can match and drop unauthorized VXLAN tunnel packets before they are decapsulated, based on information in the outer header.

  See System filters for more information.

An ACL is applied to a selected set of traffic contexts. A traffic context could be all the IPv4 or IPv6 packets arriving or leaving on a specific subinterface, or the out-of-band IP traffic arriving on a management interface, or all the in-band IPv4 or IPv6 packets that are locally terminating on the CPM of the router.

Each ACL rule, or entry, has a sequence ID. The ACL evaluates packets starting with the entry with the lowest sequence ID, progressing to the entry with the highest sequence ID. Evaluation stops at the first matching entry (that is, when the packet matches all of the conditions specified by the ACL entry).

14.1 ACL actions

When a packet matches an ACL entry, an action specified by the ACL entry is applied to the packet. An ACL entry has a primary action and an optional secondary action. The secondary action extends the primary action with additional packet handling operations.

For traffic transiting through the router, ACL entries support the following primary actions:

- accept – Allow the packet through to the next processing function.
- accept and log – Allow the packet through to the next processing function and send information about the accepted packet to the log application.
- drop – Discard the packet without ICMP generation.
- drop and log – Discard the packet without ICMP generation and send information about the dropped packet to the log application.

For traffic transiting through the router, the following secondary action is supported:

- log – Send information about the packet to the log application.

For traffic terminating on the CPM of the router, the preceding primary and secondary actions are supported, as well as the following secondary actions for ACL entries where accept is the primary action:

- distributed-policer – If the packet is extracted to the CPM, feed the packet to a hardware-based policer, which determines if the packet should be queued by the line card toward the CPM or dropped because a bit-per-second rate is exceeded.
- system-cpu-policer – If the packet has been extracted to the CPM, feed the packet to a software-based policer, which determines if the packet should be delivered to the CPM application or dropped because a packet-per-second rate is exceeded.

If a packet matches an ACL entry, no further evaluation is done for the packet. If the packet does not match any ACL entry, the default action is accept. To drop traffic that does not match any ACL entry, you can optionally configure an entry with the highest sequence ID in the ACL to drop all traffic. This causes traffic that does not match any of the lower-sequence ACL entries to be dropped.

The supported actions for each type of ACL differ based on the hardware platform where the ACL is configured. The following tables indicate which actions are supported for each ACL filter type for each hardware platform.

14.1.1 Supported ACL actions for IXR-7250 systems

Table 11: Supported actions for each ACL filter type (IXR-7250) lists the supported actions for each ACL filter type on IXR-7250 systems.

Table 11: Supported actions for each ACL filter type (IXR-7250)

ACL filter type	Action	Supported?
Interface filter (input)	accept	Yes
	accept and log	Yes
	drop	Yes
	drop and log	Yes

ACL filter type	Action	Supported?
Interface filter (output)	accept	Yes
	accept and log	Yes
	drop	Yes
	drop and log	Yes
CPM filter	accept	Yes
	accept and log	Yes
	drop	Yes
	drop and log	Yes
	accept and distributed-policer	Yes
	accept and distributed-policer and log	No log generated
	accept and system-cpu-policer	Yes
	accept and system-cpu-policer and log	No log generated
Packet capture filter	accept	Yes
	copy	Yes
System filter	accept	No
	drop	No
	drop and log	No

14.1.2 Supported ACL actions for 7220 IXR-D1, D2, and D3 systems

Table 12: Supported actions for each ACL filter type (7220 IXR-D1, D2, and D3) lists the supported actions for each ACL filter type on 7220 IXR-D1, D2, and D3 systems.

Table 12: Supported actions for each ACL filter type (7220 IXR-D1, D2, and D3)

ACL filter type	Action	Supported?
Interface filter (input)	accept	Yes
	accept and log	No log generated
	drop	Yes

ACL filter type	Action	Supported?
	drop and log	Yes, using separate CPU queue
Interface filter (output)	accept	Yes
	accept and log	No log generated
	drop	Yes
	drop and log	No log generated
CPM filter	accept	Yes
	accept and log	No log generated
	drop	Yes
	drop and log	Yes, using shared CPU queue
	accept and distributed-policer	Yes
	accept and distributed-policer and log	No log generated
	accept and system-cpu-policer	Yes
	accept and system-cpu-policer and log	No log generated
Packet capture filter	accept	Yes
	copy	Yes
System filter	accept	Yes
	drop	Yes
	drop and log	Yes

14.1.3 Supported ACL actions for 7220 IXR-H2 and H3 systems

Table 13: Supported actions for each ACL filter type (7220 IXR-H2 and H3) lists the supported actions for each ACL filter type on 7220 IXR-H2 and H3 systems.

Table 13: Supported actions for each ACL filter type (7220 IXR-H2 and H3)

ACL filter type	Action	Supported?
Interface filter (input)	accept	Yes

ACL filter type	Action	Supported?
	accept and log	No log generated
	drop	Yes
	drop and log	Yes, but logged packet is also processed by CPM filter
Interface filter (output)	accept	Yes
	accept and log	No log generated
	drop	Yes
	drop and log	No log generated
CPM filter	accept	Yes
	accept and log	No log generated
	drop	Yes
	drop and log	Yes
	accept and distributed-policer	No policing
	accept and distributed-policer and log	No policing and no log generated
	accept and system-cpu-policer	Yes
	accept and system-cpu-policer and log	No log generated
Packet capture filter	accept	Yes
	copy	Yes
System filter	accept	No
	drop	No
	drop and log	No

14.2 Interface filters

An interface filter is an IPv4 or IPv6 ACL that restricts traffic allowed to enter or exit a subinterface.

IPv4 ACLs analyze IPv4 packets. The following match criteria are supported by IPv4 ACLs:

- IPv4 destination prefix and prefix-length
- IPv4 destination address and address-mask
- TCP/UDP destination port (range)
- ICMP type/code
- IP protocol number
- IPv4 source prefix and prefix-length
- IPv4 source address and address-mask
- TCP/UDP source port (range)
- TCP flags: RST, SYN, and ACK
- Packet fragmentation: whether the packet is a fragment
- Packet fragmentation: whether the packet is a first-fragment (`fragment-offset=0` and `more-fragments=1`)

IPv6 ACLs analyze IPv6 packets. The following match criteria are supported by IPv6 ACLs:

- IPv6 destination prefix and prefix-length
- IPv6 destination address and address-mask
- TCP/UDP destination port (range)
- ICMPv6 type/code
- IPv6 next-header value. This is the value in the very first next-header field, in the fixed header.
- IPv6 source prefix and prefix-length
- IPv6 source address and address-mask
- TCP/UDP source port (range)
- TCP flags: RST, SYN, and ACK

IPv4 and IPv6 ACLs can be applied to a subinterface to restrict traffic entering or exiting that subinterface, as follows:

- Input IPv4 ACLs – When an IPv4 filter is used as an input ACL on a subinterface that carries IPv4 traffic, the rules apply to native IPv4 packets (ethertype 0x0800) that enter the subinterface and would normally terminate locally (control/management plane packets) or transit through the router.

  The rules also apply to MPLS-encapsulated IPv4 packets (ethertype 0x8847) that are terminating or transit.
- Input IPv6 ACLs – When an IPv6 filter is used as an input ACL on a subinterface that carries IPv6 traffic, the rules apply to native IPv6 packets (ethertype 0x86DD) that enter the subinterface and would normally terminate locally (control/management plane packets) or transit through the router.

  The rules also apply to MPLS-encapsulated IPv6 packets (ethertype 0x8847) that are terminating or transit.
- Output IPv4 ACLs – When an IPv4 filter is used as an output ACL on a subinterface that carries IPv4 traffic, the rules apply to native IPv4 packets (ethertype 0x0800) that exit the subinterface, including packets that originated locally (control/management plane packets) and packets that transited through the router.

  The rules do not apply to MPLS-encapsulated IPv4 packets (ethertype 0x8847) exiting the subinterface.
- Output IPv6 ACLs – When an IPv6 filter is used as an output ACL on a subinterface that carries IPv6 traffic, the rules apply to native IPv6 packets (ethertype 0x86DD) that exit the subinterface, including

packets that originated locally (control/management plane packets) and packets that transited through the router.

The rules do not apply to MPLS-encapsulated IPv6 packets (ethertype 0x8847) exiting the subinterface.

14.2.1 Creating an IPv4 ACL

The following is an example IPv4 ACL that has one entry.

Example:

This example creates an IPv4 ACL named `ip_tcp`. Within the `ip_tcp` ACL, an entry with sequence ID 1000 is configured. The action is specified as `accept`, with logging set to `true`.

The filter matches packets with IP destination address 100.1.3.1/32; for TCP traffic, if the source address 100.1.5.1/32, destination port 6789, and source port 6722 matches the filter, the traffic stream is accepted.

```
--{ * candidate shared default }--[  ]--
# acl
--{ * candidate shared default }--[ acl ]--
 ipv4-filter ip_tcp {
        entry 1000 {
            description Match_IP_Address_TCP_Protocol_Ports
            action {
                accept {
                    log true
                }
            }
            match {
                destination-address 100.1.3.1/32
                protocol tcp
                source-address 100.1.5.1/32
                destination-port {
                    value 6789
                }
                source-port {
                    value 6722
                }
            }
        }
    }
```

Example

The following is an example of an entry added to the `ip_tcp` ACL that causes all traffic to be dropped. Because it has the highest sequence ID, traffic that matches any of the lower-sequenced ACL entries would have been accepted before being evaluated by this entry. Only traffic that did not match any of the other ACL entries would be dropped by this entry.

```
--{ * candidate shared default }--[  ]--
# acl
--{ * candidate shared default }--[ acl ]--
       ipv4-filter ip_tcp {
           entry 65535 {
               action {
                   drop {
                     log true
                   }
               }
           }
```

```
        }
```

Note that the `drop` action with logging set to `true` is not supported on 7220 IXR-D1, D2, and D3 systems when it is attached as an egress filter.

14.2.2 Creating an IPv6 ACL

The following is an example IPv6 ACL that has one entry.

Example:

This example creates an IPv6 ACL named `ipv6_tcp`. Within the `ipv6_tcp` ACL, an entry with sequence ID 100 is configured. The action is specified as `accept`, with logging set to `true`.

The filter matches packets with IPv6 destination address 2001:10:1:3::1/120; for TCP traffic, if the source address 2001:10:1:5::1/120, destination port 6789, and source port 6722 matches the filter, the traffic stream is accepted.

```
--{ * candidate shared default }--[  ]--
# acl
--{ * candidate shared default }--[ acl ]--
ipv6-filter ipv6_tcp {
        entry 100 {
            description Match_Dest_Address_TCP_Src_Address_DP_SP
            action {
                accept {
                    log true
                }
            }
            match {
                destination-address 2001:10:1:3::1/120
                next-header tcp
                source-address 2001:10:1:5::1/120
                destination-port {
                    value 6789
                }
                source-port {
                    value 6722
                }
            }
        }
    }
```

14.2.3 Attaching an ACL to a subinterface

After an ACL is configured, you can attach it to a subinterface so that traffic entering or exiting the subinterface is subject to the rules defined in the ACL.

Example:

The following commands apply IPv4 and IPv6 ACLs to inbound traffic on a subinterface:

```
# interface ethernet-1/16
--{ * candidate shared default }--[ interface ethernet-1/16 ]--
# subinterface 1
--{ * candidate shared default }--[ interface ethernet-1/16  subinterface 1 ]--
# acl
--{ * candidate shared default }--[ interface ethernet-1/16 subinterface 1 acl ]--
```

```
# input ipv4-filter ip_tcp
--{ * candidate shared default }--[ interface ethernet-1/16 subinterface 1 acl ]--
# input ipv6-filter ipv6_tcp
```

Example

You can check the configuration of the interface to verify that the ACL is successfully attached. For example:

```
# info interface ethernet-1/16
interface ethernet-1/16 {
    description dut1-dut-2
    subinterface 1 {
        ipv4 {
            address 100.1.5.2/24 {
            }
            arp {
                neighbor 100.1.5.1 {
                    link-layer-address 00:00:64:01:05:05
                }
            }
        }
        ipv6 {
            address 2001:10:1:5::2/120 {
            }
            neighbor-discovery {
                neighbor 2001:10:1:5::1 {
                    link-layer-address 00:00:64:01:05:05
                }
            }
        }
        acl {
            input {
                ipv4-filter ip_tcp
                ipv6-filter ipv6_tcp
            }
        }
    }
}
```

14.2.4 Attaching an ACL to the management interface

To modulate traffic for the management interface, navigate to the subinterface of interface `mgmt0`. Under the `acl` context, attach the IPv4 or IPv6 ACL for input/output traffic.

Example:

```
--{ * candidate shared default }--[  ]--
# interface mgmt0
--{ * candidate shared default }--[ interface mgmt0 ]–
# subinterface 0
--{ * candidate shared default }--[ interface mgmt0 subinterface 0 ]--
# acl
--{ * candidate shared default }--[ interface mgmt0 subinterface 0 acl ]–
# input ipv4-filter ip_tcp
--{ * candidate shared default }--[ interface mgmt0 subinterface 0 acl ]–
# input ipv6-filter ipv6_tcp
```

Example

To verify the configuration for the management interface ACL:

```
# info interface mgmt0
   interface mgmt0     {
     admin-state enable
        subinterface 0 {
            admin-state enable
            ipv4 {
               dhcp-client true
            }
            ipv6 {
               dhcp-client true
            }
            acl {
               input {
                   ipv4-filter ip_tcp
                   ipv6-filter ipv6_tcp
               }
            }
        }
   }
```

14.2.5 Detaching an ACL from an interface

To detach an ACL from an interface, enter the subinterface context and delete the ACL from the configuration.

Example:

The following commands detach an ACL from a subinterface:

```
# interface ethernet-1/16
--{ * candidate shared default }--[ interface ethernet-1/16 ]–
# subinterface 1
--{ * candidate shared default }--[ interface ethernet-1/16 subinterface 1 ]–
# delete acl input ipv4-filter
--{ * candidate shared default }--[ interface ethernet-1/16 subinterface 1 ]--
```

Example

Use the **info interface** command to verify that the ACL is no longer part of the subinterface configuration. For example:

```
# info interface ethernet-1/16
interface ethernet-1/16 {
    description dut1-dut-2
    subinterface 1 {
        ipv4 {
            address 100.1.5.2/24 {
            }
            arp {
                neighbor 100.1.5.1 {
                    link-layer-address 00:00:64:01:05:05
                }
            }
        }
        ipv6 {
            address 2001:10:1:5::2/120 {
            }
```

```
            neighbor-discovery {
                neighbor 2001:10:1:5::1 {
                    link-layer-address 00:00:64:01:05:05
                }
            }
        }
        acl {
            input {
                ipv6-filter ipv6_tcp
            }
        }
    }
}
```

14.2.6 Detaching an ACL from the management interface

The following example detaches an ACL from the management interface:

Example:

```
--{ * candidate shared default }--[  ]--
# interface mgmt0
--{ * candidate shared default }--[ interface mgmt0 ]–
# subinterface 0
--{ * candidate shared default }--[ interface mgmt0 subinterface 0 ]--
# delete acl input ipv4-filter
```

Example

To verify that the ACL was detached from the management interface:

```
# info interface mgmt0
   interface mgmt0     {
     admin-state enable
        subinterface 0 {
            admin-state enable
            ipv4 {
               dhcp-client true
            }
            ipv6 {
               dhcp-client true
            }
            acl {
               input {
                  ipv6-filter ipv6_tcp
               }
            }
        }
   }
```

14.2.7 Modifying ACLs

You can add entries to an ACL, delete entries from an ACL, and delete the entire ACL from the configuration.

Example:

To add an entry to an ACL, enter the context for the ACL, then add the entry. For example, the following commands add an entry to IPv4 ACL `ip_tcp`:

```
--{ * candidate shared default }--[  ]--
# acl ipv4-filter ip_tcp
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# entry 65535
--{ candidate shared default }--[ acl ipv4-filter ip_tcp entry 65535 ]--
# action drop
--{ candidate shared default }--[ acl ipv4-filter ip_tcp entry 65535 action drop ]--
# log true
--{ candidate shared default }--[ acl ipv4-filter ip_tcp entry 65535 action drop ]--
```

Example

To delete an entry in an ACL, use the **delete** command under the context for the ACL and specify the sequence ID of the entry to be deleted. For example, the following commands delete the entry in IPv4 ACL `ip_tcp` with sequence ID 65535:

```
--{ candidate shared default }--[  ]--
# acl ipv4-filter ip_tcp
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# delete entry 65535
```

Example

To delete the entire ACL, use the **delete** command under the `acl` context. For example, the following commands delete the `ip_tcp` ACL:

```
# acl
--{ candidate shared default }--[ acl ]--
# delete ipv4-filter ip_tcp
```

14.2.8 Resequencing ACL entries

To aid in managing complex ACLs that have many entries, you can resequence the ACL entries to set a sequence ID number for the first entry and a constant increment for the sequence ID for subsequent entries.

For example, if you have an ACL with three entries, sequence IDs 123, 124, and 301, you can resequence the entries so that the initial entry has sequence ID 100, and the other two entries have sequence ID 110 and 120.

Example:

The following is an example of an ACL with three entries:

```
--{ candidate shared default }--[ acl ]–
# info ipv4-filter ip_tcp
 ipv4-filter ip_tcp {
        entry 123 {
            action {
                drop {
                    log true
                }
            }
```

```
            match {
                destination-address 1.1.2.1/24
            }
        }
        entry 124 {
            action {
                accept {
                    log true
                }
            }
            match {
                destination-address 1.1.3.1/24
            }
        }
        entry 301 {
            action {
                drop {
                }
            }
            match {
                destination-address 1.1.4.1/24
            }
        }
```

To resequence the entries in the ACL so that the first entry has sequence ID 100, and the next two entries are incremented by 10, enter the context for the ACL, issue the **tools resequence** command, then specify the initial sequence ID and the increment for the subsequent entries. For example:

```
--{ candidate shared default }--[  ]--
# acl ipv4-filter ip_tcp
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# tools resequence start 100 increment 10
```

After you enter the command, the ACL entries are renumbered. For example:

```
--{ candidate shared default }--[ acl ]—
# info ipv4-filter ip_tcp
 ipv4-filter ip_tcp {
        entry 100 {
            action {
                drop {
                    log true
                }
            }
            match {
                destination-address 1.1.2.1/24
            }
        }
        entry 110 {
            action {
                accept {
                    log true
                }
            }
            match {
                destination-address 1.1.3.1/24
            }
        }
        entry 120 {
            action {
                drop {
                }
            }
            match {
```

```
                destination-address 1.1.4.1/24
            }
        }
```

The **resequence** command is only available inside an ACL configuration context, and it only applies to the entries of the ACL associated with that context.

14.3 Packet capture filters

For troubleshooting purposes, the SR Linux supports ACL policies called packet capture filters. When an ingress IP packet on any line card transits through the router, and it matches a rule in a capture-filter policy, it is copied and extracted toward the CPM (using the capture-filter extraction queue) and delivered to a Linux virtual Ethernet interface, so that it can be displayed by **tcpdump** (or similar packet capture utility), or encapsulated and forwarded to a remote monitoring system.

Similarly, when an ingress IP packet on any line card terminates locally, and it matches a rule of a capture-filter policy, it is extracted toward the CPM (using the normal protocol-based extraction queue), and a header field indicates to the CPM to replicate it (after running the CPM-filter rules) toward the Linux virtual Ethernet interface.

There is one capture-filter for IPv4 traffic and another capture-filter for IPv6 traffic. The default IPv4 capture-filter policy copies no IPv4 packets and the default IPv6 capture-filter copies no IPv6 packets.

The entries for each capture-filter are installed on every line card. On the line card, the entries are evaluated after the input subinterface ACLs and before the CPM-filter ACLs. On the CPM, the entries in the capture-filter policy are evaluated after the CPM-filter entries.

When a capture-filter ACL is created, its rules are evaluated against all transit and terminating IPv4 or IPv6 traffic that is arriving on any subinterface of the router, regardless of where that traffic entered in terms of network-instance, subinterface, linecard, pipeline, and so on. Note that capture-filter ACL rules cannot override interface filter or system-filter ACL drop outcomes; packets dropped by interface filter ACLs or a system filter ACL cannot be mirrored to the control plane.

Each capture-filter entry has a set of zero or more match conditions, and one of two possible actions: accept and copy. The match conditions are the same as the other filter types. The accept action passes the matching packet to the next stage of processing, without creating a copy. The copy action creates a copy of the matching packet, extracts it toward the CPM and delivers it to the designated virtual Ethernet interface.

14.4 Control plane module (CPM) filters

For control plane protection, SR Linux supports ACL policies called CPM filters. There is one CPM filter for IPv4 traffic and another CPM filter for IPv6 traffic. When an ingress IP packet is matched by a CPM filter rule, and it is a terminating packet (that is, it must be extracted to the CPM), then it is processed according to the matching CPM filter rule.

The entries for each CPM filter are installed on every line card. They are evaluated after the input subinterface ACLs and after the capture-filter ACLs. CPM filter rules have no effect on locally originating traffic or transit traffic, and they have no interaction with output subinterface ACLs.

When a CPM filter ACL is created, its rules are evaluated against all IPv4 or IPv6 traffic that is locally terminating on the router, regardless of where that traffic entered in terms of network-instance, subinterface, linecard, pipeline, and so on.

On 7250 IXR systems, for traffic terminating on the CPM of the router, the following secondary actions are supported for ACL entries where accept is the primary action:

- distributed-policer – If the packet is extracted to the CPM, feed the packet to a hardware-based policer, which determines if the packet should be queued by the line card toward the CPM or dropped because a bit-per-second rate is exceeded.
- system-cpu-policer – If the packet has been extracted to the CPM, feed the packet to a software-based policer, which determines if the packet should be delivered to the CPM application or dropped because a packet-per-second rate is exceeded.

On 7220 IXR-D1, D2, and D3 systems, CPM filter ACLs support the following actions:

- accept – Allow the packet through to the next processing function.
- accept and distributed-policer – The packet is allowed through to the next processing function and rate limited by a policer instance implemented by the 7220 IXR-D2 and D3.
- accept and system-cpu-policer – The packet is allowed through to the next processing function and rate limited by a policer instance implemented by XDP-CPM.
- drop – Discard the packet without ICMP generation.
- drop and log – Discard the packet without ICMP generation and send information about the dropped packet to the log application.

The system-cpu-policer and distributed-policer actions police terminating traffic to ensure that the rate does not exceed a safe limit.

The system-cpu-policer action applies an aggregate rate limit, regardless of ingress line card, while the distributed-policer action applies a rate limit to the extracted traffic from each core (7250 IXR) or complex (7220 IXR) associated with the ingress port. You can have both types of policer actions in the same CPM filter entry, or only one of them.

CPM filter rules that apply a system-cpu-policer or distributed-policer action do not directly specify the policer parameters; they refer to a generically defined policer. This allows different CPM filter entries, even across multiple ACLs, to use the same policer. Optionally, each policer can be configured as entry-specific, which means a different policer instance is used by each referring filter entry, even if they are part of the same ACL.

14.5 System filters

A system filter ACL is an IPv4 or IPv6 ACL that evaluates ingress traffic before all other ACL rules. If an IP packet is dropped by a system filter rule, it is the final disposition of the packet; neither a capture-filter copy/accept action, nor an ingress interface ACL accept action, nor a CPM-filter accept action can override the drop action of a system filter.

At most one system filter can be defined for IPv4 traffic, and at most one system filter can be defined for IPv6 traffic. System filter ACLs are supported on 7220 IXR-D1, D2, and D3 systems only. They can be applied only at ingress, not egress.

When a system-filter ACL is created, its rules are automatically installed everywhere, meaning they are evaluated against all transit and terminating IPv4 or IPv6 traffic arriving on any subinterface of the router, regardless of where that traffic entered in terms of network-instance, subinterface, pipeline, and so on.

A system filter is the only type of filter that can match the outer header of tunneled packets. For VXLAN traffic, this allows you to configure a system filter that matches and drops unauthorized VXLAN tunnel packets before they are decapsulated. The system filter matches the outer header of tunneled packets; they do not filter the payload of VXLAN tunnels.

14.5.1 Creating a system filter

The following is an example of a system filter ACL that filters IPv4 traffic. When the system filter is applied, it evaluates all transit and terminating IPv4 arriving on any subinterface of the router. The system filter ACL evaluates the traffic before any other ACL filters. System filter ACLs can be configured on 7220 IXR-D1, D2, and D3 systems only.

Example:

```
--{ * candidate shared default }--[  ]--
# info acl system-filter
 acl
    system-filter {
        ipv4-filter {
            entry 44 {
                action {
                    drop {
                        log true
                    }
                }
                match {
                    source-ip {
                        address 100.1.5.1
                        mask 0.0.0.255
                    }
                }
            }
        }
    }
```

14.6 Configuring logging for ACLs

You can configure the SR Linux to log information about packets that match an ACL entry in the system log.

You can set thresholds for ACL or TCAM resource usage. When utilization of a specified resource reaches the threshold in either the rising or falling direction, it can trigger a log message.

14.6.1 Enabling syslog for the ACL subsystem

If you set the **log** parameter to **true** for the accept or drop action in an ACL entry, information about packets that match the ACL entry is recorded in the system log. You can specify settings for the log file for the ACL subsystem, including the location of the log file, maximum log file size, and the number of log files to keep.

Example:

For example, the following configuration specifies that the log file for the ACL subsystem be stored in the file `dut1_file`, located in the `/opt/srlinux/bin/logs/srbase` directory. The log file can be a maximum of 1 Mb. When the log file reaches this size, it is renamed using `dut1_file` as its base name. The five most recent log files are kept.

```
--{ candidate shared default }--[  ]--
# system
```

```
--{ candidate shared default }--[ system ]—
# info logging file dut1_file
logging {
        file dut1_file {
            directory /opt/srlinux/bin/logs/srbase/
            rotate 5
            size 1M
            subsystem acl {
            }
        }
}
```

Ensure that write permission is set for the specified directory path.

14.6.1.1 Syslog entry examples

The following are examples of syslog entries for ACLs.

IPv4 Accept:

```
acl||I Type: Ingress IPv4 Filter: testing Sequence Id: 100 Action: Accept Interface: ethernet-
1/16:1 Packet length: 56 IP Source: 100.1.5.1 Destination: 100.1.3.1 Protocol: 6 TCP Source
 port: 6722 Destination Port: 6789 Flags: SYN
```

IPv4 Drop:

```
acl||I Type: Ingress IPv4 Filter: test Sequence Id: 65535 Action: Drop Interface: ethernet-
1/16:1 Packet length: 44 IP Source: 100.3.2.3 Destination: 100.1.3.1 Protocol: 17 UDP Source
 port: 6722 Destination Port: 6789
```

IPv6 Accept:

```
acl||I Type: Ingress IPv6 Filter: tests Sequence Id: 1000 Action: Accept Interface: ethernet-
1/16:1 Packet length: 76 IP Source: 2001:10:1:5::1 Destination: 2001:10:1:3::1 Protocol: 6 TCP
 Source port: 6722 Destination Port: 6789 Flags: SYN
```

14.6.2 Logging ACL resource usage

You can set thresholds for ACL resource usage. When utilization of a specified ACL resource, such as input IPv4 filter instances, reaches the threshold in either the rising or falling direction, it can trigger a log message.

Example:

The following example sets thresholds for resource usage by input IPv4 filter instances. If the resource usage percentage falls below the `falling-threshold-log` value, a log message of priority `notice` is generated. If the resource usage percentage falls below the `rising-threshold-log` value, a log message of priority `warning` is generated.

```
--{ * candidate shared default }--[  ]--
# info platform resource-monitoring
    platform {
        resource-monitoring {
            acl {
                resource input-ipv4-filter-instances {
                    rising-threshold-log 90
                    falling-threshold-log 90
```

```
                }
            }
        }
    }
```

14.6.3 Logging TCAM resource usage

You can set thresholds for TCAM resource usage. When utilization of a specified TCAM resource, such as TCAM used by IPv4 CPM filters, reaches the threshold in either the rising or falling direction, it can trigger a log message.

Example:

The following example sets thresholds for TCAM resource usage by IPv4 CPM filters. If the resource usage percentage falls below the `falling-threshold-log` value, a log message of priority `notice` is generated. If the resource usage percentage falls below the `rising-threshold-log` value, a log message of priority `warning` is generated.

```
--{ * candidate shared default }--[  ]--
# info platform resource-monitoring
    platform {
        resource-monitoring {
            tcam {
                resource cpm-capture-ipv4 {
                    rising-threshold-log 90
                    falling-threshold-log 90
                }
            }
        }
    }
```

14.7 Collecting and displaying ACL statistics

The SR Linux can collect statistics for packets matching an ACL and display statistics for the matched packets. You can display the amount of system resources (TCAM) used by each type of ACL on each line card. ACL statistics can also be displayed using **show** commands.

14.7.1 Collecting ACL statistics

You can configure an ACL to collect statistics for packets matching the ACL. Statistics can be collected for packets that match each ACL entry, as well as for matching input/output traffic per subinterface.

Example:

The following example configures the ACL to record the number of matching packets for each entry:

```
--{ candidate shared default }--[ acl ]--
# ipv4-filter ip_tcp
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# statistics-per-entry true
```

Example

By default, if two or more subinterfaces on the same line card reference the same ACL for filtering the same direction of traffic, they use a shared instance of the same ACL in hardware. This means that per-entry statistics (including the number of matched packets and the time stamp of the last matching packet), if enabled, reflect the aggregate of the data gathered for the multiple subinterfaces.

To collect per-entry, per-subinterface statistics, instead of the aggregate of the subinterfaces where the ACL is applied, you can configure an ACL to operate in subinterface-specific mode.

If you change an ACL from subinterface-specific mode to shared mode, or the other way around, during the transition from one mode to the next, traffic continues to be subject to the previous mode until the system resources (TCAM) entries are programmed for the new mode.

The following example configures the ACL to collect statistics for matching packets inbound and outbound on each subinterface:

```
--{ candidate shared default }--[ acl ]--
# ipv4-filter ip_tcp
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# subinterface-specific input-and-output
```

You can configure the following values for the **subinterface-specific** parameter:

- **disabled** (the default) – All subinterfaces on a single line card that reference the ACL as an input ACL use a shared filter instance, and all subinterfaces on a single line card that reference the ACL as an output ACL use a shared filter instance.
- **input-only** – All subinterfaces on a single line card that reference the ACL as an output ACL use a shared filter instance, but each subinterface that references the ACL as an input ACL uses its own separate instance of the filter.
- **output-only** – All subinterfaces on a single line card that reference the ACL as an input ACL use a shared filter instance, but each subinterface that references the ACL as an output ACL uses its own separate instance of the filter.
- **input-and-output** – Each subinterface that references the ACL as either an input ACL or an output ACL uses its own separate instance of the filter.

14.7.2 Displaying ACL statistics

Use the **info from state** command to display the matched packet statistics and the time of the last match for the interfaces to which the ACL is attached.

Example:

In the following example, the ACL is attached to two interfaces, and statistics are collected for each subinterface:

```
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# info from state
    subinterface-specific input-and-output
    statistics-per-entry true
    entry 1000 {
        description Match_IP_Address_TCP_Protocol_Ports
        action {
            accept {
                log true
            }
        }
```

```
        match {
            destination-address 100.1.3.1/32
            protocol tcp
            source-address 100.1.5.1/32
            destination-port {
                value 6789
            }
            source-port {
                value 6722
            }
        }
        statistics {
            aggregate {
                in-matched-packets 3000
                in-last-match 2019-07-16T10:53:00.1563Z
                out-matched-packets 0
            }
            per-interface {
                subinterface ethernet-1/16.1 {
                    in-matched-packets 3000
                    in-last-match 2019-07-16T10:53:00.1563Z
                }
            }
        }
    }
    entry 65535 {
        action {
            drop {
                log true
            }
        }
        statistics {
            aggregate {
                in-matched-packets 1000
                in-last-match 2019-07-16T10:53:30.1563Z
            }
            per-interface {
                subinterface ethernet-1/16.1 {
                    in-matched-packets 1000
                    in-last-match 2019-07-16T10:53:30.1563Z
                }
            }
        }
    }
```

If the match criteria changes for an ACL entry, the statistics counter does not reset to zero. To reset the statistics counter for an ACL entry to zero, use the **tools acl clear** command, as described in Clearing ACL statistics.

14.7.3 Displaying ACL resource usage

Use the **info from state platform** command to display the amount of system resources (TCAM) used by each type of ACL on each line card.

Example:

The following example displays the TCAM usage for input IPv4 ACLs on a line card:

```
--{ candidate shared default }--[  ]--
# info from state platform linecard 1 forwarding-complex 0 tcam resource if-input-ipv4
    platform {
        linecard 1 {
```

```
            forwarding-complex 0 {
                tcam {
                    resource if-input-ipv4 {
                        free 4094
                        reserved 2
                        programmed 2
                    }
                }
            }
        }
    }
```

Example

The following example displays the amount of system resources allocated to input IPv4 ACLs on a line card, and how much is used and free:

```
--{ candidate shared default }--[  ]--
# info from state platform linecard 1 forwarding-complex 0 acl resource input-ipv4-filter-
instances
    platform {
        linecard 1 {
            forwarding-complex 0 {
                acl {
                    resource input-ipv4-filter-instances {
                        used 1
                        free 254
                    }
                }
            }
        }
    }
```

14.7.4 Clearing ACL statistics

To reset ACL statistics counters to zero, use the **tools acl clear** command. This command can clear statistics at the IPv4 / IPv6 / CPM filter level, ACL entry level, or for an interface or subinterface to which the ACL is attached.

Example:

The following example clears statistics for an IPv4 filter:

```
--{ candidate shared default }--[ acl ]--
# tools acl ipv4-filter tcp_ip clear
```

Example

After this command is executed, the **info from state** output for the IPv4 filter includes a timestamp indicating when the statistics were cleared. For example:

```
--{ candidate shared default }--[ acl ipv4-filter tcp_ip ]--
# info from state
    subinterface-specific output-only
    statistics-per-entry true
    entry 1000 {
        description Match_IP_Address_TCP_Protocol_Ports
        action {
            accept {
                log true
```

```
            }
        }
        match {
            destination-address 100.1.3.1/32
            protocol tcp
            source-address 100.1.5.1/32
            destination-port {
                value 6789
            }
            source-port {
                value 6722
            }
        }
        statistics {
            last-clear 2019-10-03T13:53:51.000Z
        }
    }
```

Example

The following example clears statistics for a specific entry in the IPv4 filter:

```
--{ candidate shared default }--[ acl ]--
# tools acl ipv4-filter tcp_ip entry 1000 statistics clear
```

Example

The following example clears statistics for a specified subinterface for a specified entry in the IPv4 filter:

```
--{ candidate shared default }--[ acl ]--
# tools acl ipv4-filter tcp_ip entry 1000 statistics per-interface subinterface 1 clear
```

14.7.5 Displaying ACL statistics using show commands

You can display ACL statistics using relevant **show** commands.

Example:

To display information about all active ACLs, use the **show acl summary** command. For example:

```
# show acl summary
-------------------------------------------------------------------------------
IPv4 Filter ACLs
-------------------------------------------------------------------------------
Filter : ip_tcp
Active on : 1 subinterfaces (input) and 0 subinterfaces (output)
Entries : 2
-------------------------------------------------------------------------------
IPv6 Filter ACLs
-------------------------------------------------------------------------------
Filter : ipv6_tcp
Active on : 1 subinterfaces (input) and 0 subinterfaces (output)
Entries : 1
-------------------------------------------------------------------------------
```

Example

You can display statistics for a specific ACL, including how many times each ACL entry was matched on all subinterfaces to which the ACL was applied. For example:

```
--{ candidate shared default }--[  ]--
# show acl ipv4-filter ip_tcp
============================================================================================
Filter        : ip_tcp
SubIf-Specific: input-and-output
Entry-stats   : yes
Entries       : 2
--------------------------------------------------------------------------------------------
 Subinterface     Input   Output
ethernet-1/16.1   yes     no
--------------------------------------------------------------------------------------------
Entry 1000
  Match               : protocol=tcp, 100.1.5.1/32(6722-6722)->100.1.3.1/32(6789-
6789)
  Action              : accept
  Input Match Packets : 3000
  Input Last Match    : 18 seconds ago
  Output Match Packets: 0
  Output Last Match   : never
Entry 65535
  Match               : protocol=<undefined>, any(*)->any(*)
  Action              : drop
  Input Match Packets : 1000
  Input Last Match    : 6 minutes ago
  Output Match Packets: 0
  Output Last Match   : never
```

Example

To display per-interface statistics for packets matching each ACL entry, specify the interface name in addition to the ACL name. For example:

```
--{ candidate shared default }--[  ]--
# show acl ipv4-filter ip_tcp interface ethernet-1/16
============================================================================================
Filter        : ip_tcp
SubIf-Specific: input-and-output
Entry-stats   : yes
Entries       : 2
--------------------------------------------------------------------------------------------
 Subinterface     Input   Output
ethernet-1/16.1   yes     no
--------------------------------------------------------------------------------------------
Entry 1000
  Match               : protocol=tcp, 100.1.5.1/32(6722-6722)->100.1.3.1/32(6789-
6789)
  Action              : accept
  Input Match Packets : 3000
  Input Last Match    : 5 minutes ago
  Output Match Packets: 0
  Output Last Match   : never
Entry 65535
  Match               : protocol=<undefined>, any(*)->any(*)
  Action              : drop
  Input Match Packets : 1000
  Input Last Match    : 10 minutes ago
  Output Match Packets: 0
  Output Last Match   : never
```

Example

To display statistics for packets matching a CPM filter, specify the CPM filter type (IPv4 or IPv6). For example:

```
--{ candidate shared default }--[  ]--
# show acl cpm-filter ipv4-filter
===============================================================================
Filter     : CPM IPv4-filter
Entry-stats: no
Entries    : 1
-------------------------------------------------------------------------------
Entry 1001
  Match          : protocol=<undefined>, any(*)->1.1.2.1/24(*)
  Action         : none
  Matched Packets: 0
-------------------------------------------------------------------------------
```

15 SR Linux applications

The SR Linux is a suite of modular, lightweight applications running like any others in a Linux environment. Each SR Linux application supports a different protocol or function, such as BGP, LLDP, AAA, and so on. These applications use gRPC and APIs to communicate with each other and external systems over TCP.

One SR Linux application, the application manager (app_mgr), is itself responsible for monitoring the health of the process IDs running each SR Linux applications, and restarting them if they fail. The application manager reads in application-specific YAML configuration and YANG models, and starts each application (or allows an application not to start if there no configuration exists for it). There is an instance of the app_mgr that handles applications running on the CPM, and an instance of the app_mgr on each IMM that handles applications running on the line card.

In addition to the Nokia-provided SR Linux applications, the SR Linux supports installation of user-defined applications, which are managed and configured in the same way as the default SR Linux applications.

15.1 Installing an application

About this task

To install an application, copy the application files into the appropriate SR Linux directories, then reload the application manager and start the application.

The example in this topic installs an application called `fib_agent`. The application consists of files named `fib_agent.yml`, `fib_agent.sh`, `fib_agent.py`, and `fib_agent.yang`. The `fib_agent.yml` file is installed in the `/etc/opt/srlinux/appmgr/` directory. The `.yml` file for a user-defined application must reside in this directory in order for the app_mgr to read its YAML configuration.

The `.yml` file defines the locations of the other application files. The other application files can reside anywhere in the system other than in the `/opt/srlinux/` directory or any tempfs file system.

In this example, the `fib_agent.sh` and `fib_agent.py` files are installed in the directory`/user_agents/`, and the `fib_agent.yang` file is installed in the directory`/yang/`. The locations for these files are defined in the `fib_agent.yml` file.

Procedure

Step 1. Copy the application files into the SR Linux directories.

Example

```
# cp fib_agent.yml /etc/opt/srlinux/appmgr/.
# cp fib_agent.sh /user_agents/.
# cp fib_agent.py /user_agents/.
# cp fib_agent.yang /yang/.
```

Step 2. From the SR Linux CLI, reload the application manager.

Example

```
--{ candidate }--[ ]--
# tools system app-management application app_mgr reload
```

Step 3. Apply the changes to the configuration.

Example

```
--{ candidate }--[  ]--
# fib-agent
--{ candidate }--[ fib-agent ]--
# commit stay
All changes have been committed. Starting new transaction.
--{ candidate }--[ fib-agent ]--
```

Step 4. Verify that the application is running.

Example

```
# show system application fib_agent
  +-----------+-----+---------+----------------------+--------------------------+
  |  Name     | PID |  State  |        Version       |       Last Change        |
  +===========+=====+=========+======================+==========================+
  | fib_agent | 227 | running | v21.3.0-61-gd19567393 | 2021-01-13T20:16:45.697Z |
  +-----------+-----+---------+----------------------+--------------------------+
```

15.2 Starting an application

To start an SR Linux application instance, use the **start** option in the **tools system app-management** command. To terminate a running application instance and restart it, use the **restart** option.

Example:

To start an SR Linux application instance:

```
# tools system app-management application mpls_mgr start
/system/app-management/application[name=mpls_mgr]:
    Application 'mpls_mgr' was started
```

Example

To restart an SR Linux application instance:

```
# tools system app-management application mpls_mgr restart
/system/app-management/application[name=mpls_mgr]:
    Application 'mpls_mgr' was killed with signal 9
/system/app-management/application[name=mpls_mgr]:
    Application 'mpls_mgr' was restarted
```

15.3 Terminating an application

You can use the **stop**, **quit**, or **kill** options in the **tools system app-management** command to terminate an SR Linux application.

- **stop** gracefully terminates the application, allowing it to clean up before exiting.
- **quit** terminates the application and generates a core dump. The core dump files are saved in the `/var/log/srlinux/cores/` directory.
- **kill** terminates the application immediately, without allowing it to clean up before exiting.

Example:

To terminate an application gracefully:

```
# tools system app-management application mpls_mgr stop
/system/app-management/application[name=mpls_mgr]:
    Application 'mpls_mgr' was killed with signal 15
```

Example

To terminate an application and generate a core dump:

```
# tools system app-management application mpls_mgr quit
/system/app-management/application[name=mpls_mgr]:
    Application 'mpls_mgr' was killed with signal 3
```

Example

To terminate an application immediately:

```
# tools system app-management application mpls_mgr kill
/system/app-management/application[name=mpls_mgr]:
    Application 'mpls_mgr' was killed with signal 9
```

15.4 Reloading application configuration

Reloading an application causes the app_mgr to reread the application's YAML configuration and restart the application using settings in its YAML file.

Example:

To reload the configuration of the app_mgr application:

```
--{ * running }--[  ]--
# tools system app-management application app_mgr reload
--{ * running }--[  ]--
```

15.5 Clearing application statistics

You can display statistics collected for an application with the **info from state** command. To reset the statistics counters for the application, use the **statistics clear** option in the **tools system app-management** command.

Example:

To reset the statistics counters for an application:

```
# tools system app-management application mpls_mgr statistics clear
```

15.6 Restricted operations for applications

An application may have one or more operations that are restricted by default. For example, the linux_mgr application has stop, quit, and kill as restricted operations, meaning that these options are not available when entering the **tools system app-management** command for the linux_mgr application.

Table 14: Restricted operations for SR Linux applications lists the restricted operations for each SR Linux application.

Table 14: Restricted operations for SR Linux applications

Application	Restricted operations
aaa_mgr	reload
acl_mgr	reload
app_mgr	start, stop, restart, quit, kill
arp_nd_mgr	reload
bfd_mgr	reload
bgp_mgr	reload
chassis_mgr	stop, quit, kill, reload
device_mgr	reload
dhcp_client_mgr	stop, reload
fib_mgr	reload
gnmi_server	reload
idb_server	start, stop, restart, quit, kill, reload
json_rpc_config	reload
linux_mgr	stop, quit, kill
lldp_mgr	reload
log_mgr	reload
mgmt_server	start, stop, quit, kill, reload
mpls_mgr	reload
net_inst_mgr	start, stop, quit, kill, reload
oam_mgr	reload

Application	Restricted operations
plcy_mgr	reload
qos_mgr	reload
sdk_mgr	reload
static_route_mgr	reload
supportd	reload
xdp_cpm	stop, quit, kill, reload
xdp_lc	reload

Restricted options are specified in the `restricted-operations` setting in the YAML file for the application.

15.7 Configuring an application

About this task

To configure an SR Linux application, edit settings in the application YAML file, then reload the application manager to activate the changes.

The example in this section shows how to configure an application to specify the action the SR Linux device takes when the application fails. If an SR Linux application fails a specified number of times over a specified time period, the SR Linux device can reboot the system or attempt to restart the application after waiting a specified number of seconds.

For example, if the aaa_mgr application crashes 5 times within a 500-second window, the SR Linux device can be configured to wait 100 seconds, then restart the aaa_mgr application.

The following actions can be taken if an SR Linux application fails:

- Reboot the system
- Wait a specified number of seconds, then attempt to restart the failed application
- Move the failed application to error state without rebooting the system or attempting to restart the application

If you stop or restart an application using the **tools system app-management** command in the SR Linux CLI, it is not considered an application failure; the failure action for the application, if one is configured, does not occur. However, if the failed application waits a specified period of time (or forever) to be restarted, or has been moved into error state, you can restart the application manually with the **tools system app-management application restart** CLI command.

To configure the failure action for an application:

Procedure

Step 1. Check the status of the SR Linux applications:

Example

```
# show system application
+-------------+-----+---------+----------------------+--------------------------+
|    Name     | PID |  State  |        Version       |       Last Change        |
+=============+=====+=========+======================+==========================+
| aaa_mgr     | 242 | error   | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.529Z |
| acl_mgr     | 261 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.530Z |
| app_mgr     | 185 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.585Z |
| arp_nd_mgr  | 270 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.530Z |
| bfd_mgr     | 279 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.530Z |
| bgp_mgr     | 850 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.823Z |
| chassis_mgr | 288 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.530Z |
| dev_mgr     | 194 | running |                       | 2021-03-05T20:34:48.803Z |
| fib_mgr     | 311 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.531Z |
| gnmi_server | 857 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.826Z |
| idb_server  | 229 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.033Z |
| json_rpc    | 864 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.828Z |
| linux_mgr   | 320 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.531Z |
| lldp_mgr    | 872 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.838Z |
| log_mgr     | 330 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.532Z |
| mgmt_server | 340 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.532Z |
| mpls_mgr    | 357 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.532Z |
| net_inst_mgr| 377 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.532Z |
| oam_mgr     | 392 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.533Z |
| plcy_mgr    | 401 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.533Z |
| qos_mgr     | 842 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.750Z |
| sdk_mgr     | 418 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.533Z |
| sshd-mgmt   | 107 | running |                       | 2021-03-05T20:34:53.701Z |
| supportd    | 480 | running |                       | 2021-03-05T20:34:49.534Z |
| xdp_cpm     | 520 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.534Z |
| xdp_lc_1    | 539 | running | v21.3.0-61-gd19567393 | 2021-03-05T20:34:49.535Z |
+-------------+-----+---------+----------------------+--------------------------+
```

Step 2. Use the **info from state** command to check the current failure action settings for the application to configure. These settings are highlighted in the following example:

Example

```
# info from state system app-management application aaa_mgr
    system {
        app-management {
            application aaa_mgr {
                pid 242
                state error
                last-change 2021-03-05T20:34:49.529Z
                author Nokia
                failure-threshold 3
                failure-window 300
                failure-action reboot
                path /opt/srlinux/bin
                launch-command ./sr_aaa_mgr
                search-command ./sr_aaa_mgr
                version v21.3.0-61-gd19567393
                restricted-operations [
                    reload
                ]
                statistics {
                    restart-count 0
                }
                yang {
```

```
                    modules [
                        srl_nokia-aaa
                        srl_nokia-aaa-types
                    ]
                    source-directories [
                        /opt/srlinux/models/ietf
                        /opt/srlinux/models/srl_nokia/models/common
                        /opt/srlinux/models/srl_nokia/models/network-instance
                        /opt/srlinux/models/srl_nokia/models/system
                    ]
                }
            }
        }
    }
```

The following failure action settings can be configured for an application:

- `failure-threshold`: number of times that the application must fail during the *failure-window* period before the *failure-action* is taken; the default is three times.
- `failure-window`: number of seconds over which the application must fail the *failure-threshold* number of times before the *failure-action* is taken; the default is 300 seconds.
- `failure-action`: action to take if the application fails *failure-threshold* times over *failure-window* seconds. This can be one of the following:
  - `reboot`: reboot the system; this is the default *failure-action*.
  - `wait=`*`seconds`*: wait this number of seconds, then attempt to restart the application.
  - `wait=forever`: move the application to error state and do not reboot the system or attempt to restart the application.

Step 3. Edit the YAML configuration for the application.

The YAML configuration files for SR Linux applications are located in the directory `/opt/srlinux/appmgr` on the SR Linux device. They are named `sr_`*`application_name`*`_config.yml`; for example, `sr_aaa_mgr_config.yml`.

Step 4. In the `.yml` file, add or change the settings for the `failure-threshold`, `failure-window`, and `failure-action` parameters.

Example

In the following example, the failure action settings in the `sr_aaa_mgr_config.yml` file are configured so that if the aaa_mgr application fails 5 times over a 500-second period, the SR Linux device waits 100 seconds, then attempts to restart the aaa_mgr application:

```
aaa_mgr_setup:
   path: /opt/srlinux/bin
   launch-command: ./aaamgr_set_env.sh
   run-as-user: root
   never-show: Yes
   never-restart: Yes
   start-order: 1
aaa_mgr:
   path: /opt/srlinux/bin
   launch-command:  ./sr_aaa_mgr
   search-command: ./sr_aaa_mgr
   run-as-user: root
   restricted-operations: ['reload']
   failure-threshold: 5
   failure-window : 500
   failure-action: "wait=100"
```

```
    yang-modules:
        names:
            - "srl_nokia-aaa"
            - "srl_nokia-aaa-types"
        source-directories:
            - "/opt/srlinux/models/ietf"
            - "/opt/srlinux/models/srl_nokia/models/common"
            - "/opt/srlinux/models/srl_nokia/models/system"
            - "/opt/srlinux/models/srl_nokia/models/network-instance"
```

Step 5. Save and close the `.yml` configuration file.

Step 6. In the SR Linux CLI, reload the application manager:

Example

```
# tools system app-management application app_mgr reload
```

This command reloads any application whose `.yml` configuration file has changed. It does not affect any service.

Step 7. Use the **info from state** command to verify that the changes to the failure action settings are now in effect.

Example

```
# info from state system app-management application aaa_mgr
    system {
        app-management {
            application aaa_mgr {
                pid 242
                state running
                last-change 2021-03-05T20:44:31.403Z
                author Nokia
                failure-threshold 5
                failure-window 500
                failure-action wait=100
                path /opt/srlinux/bin
                launch-command ./sr_aaa_mgr
                search-command ./sr_aaa_mgr
                version v21.3.0-61-gd19567393
                restricted-operations [
                    reload
                ]
                statistics {
                    restart-count 0
                }
                yang {
                    modules [
                        srl_nokia-aaa
                        srl_nokia-aaa-types
                    ]
                    source-directories [
                        /opt/srlinux/models/ietf
                        /opt/srlinux/models/srl_nokia/models/common
                        /opt/srlinux/models/srl_nokia/models/network-instance
                        /opt/srlinux/models/srl_nokia/models/system
                    ]
                }
            }
        }
    }
```

15.8 Partioning and isolating application resources

The SR Linux protects system processes through the use of control groups (cgroups), which impose resource consumption limits on resource-intensive customer applications.

15.8.1 Cgroup profiles

Cgroup profiles define how usage limits are applied. On the SR Linux, cgroup profiles are supported for CPU and memory and are defined in cgroup_profile.json configuration files.

SR Linux provides a default cgroup profile; customers can configure additional cgroup profiles.

15.8.1.1 Default cgroup profile

The SR Linux-provided default cgroup profile is located in the `/opt/srlinux/appmgr/cgroup_profile.json` directory.

Note: Editing the default cgroup profile is not recommended.

If the default cgroup profile fails to parse or be read by the app_mgr, the SR Linux does not start.

The default cgroup_profile.json file definition is shown below:

```
{
 "profiles": [
  {
   "name": "workload.primary",
   "path": "workload.slice/primary",
   "controller": {
    "memory": {
     "max": 0.8,
     "swap_max": 0,
     "low": 0
    },
    "cpu": {
     "weight": "10000",
     "period": "100000",
     "quota": "0"
    },
    "cpuset": {
     "cpus": "",
     "mems": ""
    }
   }
  },
  {
   "name": "workload.datapath",
   "path": "workload.slice/datapath",
   "controller": {
    "memory": {
     "max": 0.8,
     "swap_max": 0,
     "low": 0
    },
    "cpu": {
     "weight": "10000",
```

```
     "period": "100000",
     "quota": "0"
    },
    "cpuset": {
     "cpus": "all",
     "mems": ""
    }
   }
  },
  {
   "name": "workload.secondary",
   "path": "workload.slice/secondary",
   "controller": {
    "memory": {
     "max": 0.5,
     "swap_max": 0,
     "low": 0
    },
    "cpu": {
     "weight": "10000",
     "period": "100000",
     "quota": "0"
    },
    "cpuset": {
     "cpus": "",
     "mems": ""
    }
   }
  },
  {
   "name": "user.default",
   "path": "user.slice/default",
   "controller": {
    "memory": {
     "max": 0.25,
     "swap_max": 0,
     "low": 0
    },
    "cpu": {
     "weight": "1000",
     "period": "100000",
     "quota": "0"
    },
    "cpuset": {
     "cpus": "",
     "mems": ""
    }
   }
  }
 ]
}
```

Table 15: Default cgroup profile parameters describes the default cgroup profile parameters.

Table 15: Default cgroup profile parameters

Parameter	Description
name	The cgroup profile name. Type: string

Parameter	Description
path	The cgroup directory path relative to a unified root path. A typical unified root path is /sys/fs/cgroup or /mnt/cgroup/unified Type: string
controller	The memory controller configuration. **max** This number denotes the percentage of total memory. The actual memory value is calculated as (max. × total_memory) and is set in the memory.max interface file of the cgroup. If the value is 0, this configuration is ignored. The range is from 0 to 1, the default is 0.8. **low** This number denotes the percentage of total memory. The actual memory value is calculated as (max. × total_memory) and is set in the memory.low interface file of the cgroup. The range is from 0 to 1, the default is 0.8.
cpu	The CPU controller configuration. **weight** This value is set in the cpu.weight interface file of the cgroup. The range is from 1 to 10 000, the default is 100. **period** This value specifies a period of time, in microseconds, for how regularly a cgroup's access to CPU resources should be reallocated.This value is set in the cpu.max interface file of the cgroup. The range is from 1000 to 1 000 000, the default is 100 000. **quota** This value specifies the total length of time, in microseconds, for which all tasks in a cgroup can run during one period (as defined in the **period** parameter). If quota is set to 0, it translates to "max" in the cpu.max interface file. The range is from 1000 to 1 000 000, the default is max.
cpuset	CPU usage information for the cgroup. **cpus** This value indicates the CPUs used by the cgroup. This can be `""`, meaning use all CPUs except for the isolated CPUs; this is the default. The value `all` means include the isolated CPUs for cgroup usage. The value `x, y-z`, where `x`, `y`, and `z` are CPU numbers, means use a specific CPU or a range of CPUs. **mems** This value is used for scheduling multiple NUMA (non-uniform memory access) aware applications in the cgroup.

15.8.1.2 Customer-defined cgroup profile

Customers can configure cgroup profiles in the `/etc/opt/srlinux/appmgr/cgroup_profile.json` directory. The app_mgr creates this directory at boot up if it does not exist.

If a customer-defined cgroup profile fails to load, the system continues to function and applications that are loaded into the customer cgroup are loaded using Nokia defaults, listed below.

- Nokia-written applications run in the workload.slice/primary cgroup along with any processes that are started by linuxadmin, including sr_cli.

- Non-Nokia-written applications run in the workload.slice/secondary cgroup. If a customer builds an application and launches it using the app_mgr without specifying a cgroup, the application runs in this cgroup
- All interactive user applications run in the user.slice/default cgroup, including sr_cli when not started by linuxadmin.

The admin user is treated as any other user in the system. Its processes fall into the user.slice/default cgroup.

15.8.2 Configuring a cgroup

Customers can configure up to three cgroups in the `/etc/opt/srlinux/appmgr/cgroup_profile.json` directory. Customer applications are assigned to these groups. Any more than three configured cgroups are ignored. The depth of cgroups is limited to two levels where, for example, workload is one level, and primary/secondary are two levels. Any levels beyond this are also ignored.

If a cgroup with the same name is used in multiple customer-defined profiles, the system ignores it and uses the cgroup defined in the default profile.

15.8.2.1 Cgroup configuration example

About this task

The following example shows the configuration of two customer-defined cgroups: one for a lightweight database that needs priority access to resources, and one for storing the users of the database.

The steps and the outputs are described below.

The configuration above created two cgroup profiles: one for the database slice and one for the frontend slice. The profile for the database slice is configured to limit the database to 50% of system memory. The profile for the frontend slice is configured to limit the web front end to 20% of system memory.

In addition, both cgroup profiles are configured to limit CPU resources for their respective cgroup. The database server CPU is weighted at 10000 (the maximum CPU weight) and the frontend server CPU is weighted at 5000 (half the CPU weight of the database). The weights are added together and each group is allocated its ratio of CPU as a proportion of the sum. The periods are kept the same, and no guaranteed CPU is granted.

Procedure

Step 1. Install the application, including the YAML binary file and optional YANG module.

In this example, YAML defines two applications: dtw-database and dtw-frontend. These applications are placed into their own cgroups: distributetheweb.slice/database and distributetheweb.slice/frontend.

The app-mgr creates the cgroups.

The output below shows the installation of the database and a web front end.

```
dtw-database:
   path: /opt/distributetheweb/bin/
   launch-command: ./run_db
   search-command: ./run_db
   failure-threshold: 100
   failure-action: "wait=60"
   cgroup: distributetheweb.slice/database
   oom-score-adj: -500
```

```
   yang-modules:
       names:
           - "database"
       source-directories:
           - "/opt/distributetheweb/yang/"
```

```
dtw-frontend:
   path: /opt/distributetheweb/bin/
   launch-command: ./run_frontend
   search-command: ./run_frontend
   failure-threshold: 100
   failure-action: "wait=60"
   cgroup: distributetheweb.slice/frontend
   oom-score-adj: 200
   yang-modules:
       names:
           - "frontend"
       source-directories:
           - "/opt/distributetheweb/yang/"
```

Step 2. Configure the cgroup profiles in the `/etc/opt/srlinux/appmgr/cgroup_profile.json` directory.

The output below shows the configuration.

```
{
 "profiles": [
  {
   "name": "distributetheweb.database",
   "path": "distributetheweb.slice/database",
   "controller": {
    "memory": {
     "max": 0.5,
     "swap_max": 0,
     "low": 0
    },
    "cpu": {
     "weight": "10000",
     "period": "100000",
     "quota": "0"
    }
   }
  },
  {
   "name": "distributetheweb.frontend",
   "path": "distributetheweb.slice/frontend",
   "controller": {
    "memory": {
     "max": 0.2,
     "swap_max": 0,
     "low": 0
    },
    "cpu": {
     "weight": "5000",
     "period": "100000",
     "quota": "0"
    }
   }
  }
 ]
}
```

15.8.3 Kernel low-memory killer

The kernel low-memory killer driver monitors the memory state of a running system. It reacts to high memory pressure by killing the least essential processes to keep the system performing at acceptable levels.

When the system is low in memory and cannot find free memory space, the out_of_memory function is called. The out_of_memory function makes memory available by killing one or more processes.

When an out-of-memory (OOM) failure occurs, the out_of_memory function is called and it obtains a score from the select_bad_process function. The process with the highest score is the one that is killed. Some of the criteria used to identify a bad process include the following:

- The kernel needs a minimum amount of memory for itself.
- Try to reclaim a large amount of memory.
- Do not kill a process that is using a small amount of memory.
- Try to kill the minimum number of processes.
- Algorithms that elevate the sacrifice priority on processes the user wants to kill.

In addition to this list, the OOM killer checks the out-of-memory (OOM) score. The OOM killer sets the OOM score for each process and then multiplies that value by memory usage. The processes with higher values have a high probability of being terminated by the OOM killer.

15.8.3.1 SR Linux process kill strategy

The kernel calculates the oom_score using the formula 10 × percentage of memory used by the process. The maximum score is 10 × 100% = 1000. The oom_score of a process can be found in the `/proc` directory (`/proc/$pid/oom_score`). An oom_score of 1000 means the process is using all the memory, an oom_score of 500 means it is using half the memory, and an oom_score of 0 means it is using no memory.

The OOM killer checks the `oom_score_adj` file in the `/proc/$pid/oom_score_adj` directory to adjust its final calculated score. The default value is 0.

The `oom_score_adj` value can range from -1000 to 1000. A score of -1000 results in a process using 100% of the memory and in not being terminated by the OOM killer. However, a score of 1000 causes the Linux kernel to terminate the process even when it uses minimal memory. A score of -100 results in a process using 10% of the memory before it is considered for termination, as its score remains 0 until its unadjusted score reaches 100.

The `oom_score_adj` value for each process is defined in its corresponding YAML definition file. The system groups the processes based on their score, which the SR Linux OOM killer uses as the hierarchy for terminating a rogue process, as follows:

- group1 = -998, for processes that should never be killed, such as app_mgr, idb_server, and all processes on the IMM.
- group2 = -200, for processes that ideally should not be killed because doing so has a substantial impact on the system, such as mgmt_server, chassis_mgr, and net_inst_mgr. A score of -200 means the process gets to use 20% of the memory before their memory use starts being counted.
- group3 = 0, for process that should be killed if they are using too much memory such as BGP, ISIS, and OSPF.
- group4 = 500, for processes that should be preferentially killed, such as cli and oam_mgr

Table 16: OOM adjust score per process lists the OOM adjust score for each process.

Table 16: OOM adjust score per process

Process name	OOM adjust score
aaa_mgr	0
acl_mgr	0
app_mgr	-998
sarp_nd_mgr	-200
bfd_mgr	-200
bgp_mgr	0
chassis_mgr	-200
dev_mgr	-200
dhcp_client_mgr	0
dhcp_relay_mgr	0
eth_switch_mgr	-200
evpn_mgr	0
fib_mgr	0
gnmi_server	500
idb_server	-998
isis_mgr	0
json_rpc	500
l2_mac_learn_mgr	0
l2_mac_mgr	0
l2_static_mac_mgr	0
lag_mgr	0
linux_mgr	0
lldp_mgr	0
log_mgr	0

Process name	OOM adjust score
mcid_mgr	0
mgmt_server	-200
mpls_mgr	0
net_inst_mgr	-200
oam_mgr	500
ospf_mgr	0
plcy_mgr	0
qos_mgr	0
sdk_mgr	500
sflow_sample_mgr	500
sshd-mgmt	0
static_route_mgr	0
supportd	0
timesrv	0
vrrp_mgr	0
vxlan_mgr	0
xdp_cpm	-200

15.8.4 Application manager extensions for cgroups

The cgroup feature provides two additional parameters in the app_mgr YAML file, the **cgroup** parameter and the **oom-score-adj** parameter.

The app_mgr uses the **cgroup** parameter to launch an application within a specific cgroup. A valid value for the **cgroup** parameter is the path of a cgroup as specified in the cgroup profile (equivalent to `/profiles/name[]/path`), from the cgroupv2 root. If this cgroup does not exist, the app_mgr launches the user application from the default cgroup profile path `workload.slice/secondary`.

The app_mgr uses the **oom-score-adj** parameter to set the out-of-memory adjust score for a process. This score is fed into the SR Linux OOM killer. Valid **oom-score-adj** scores are any value in the range of -1000 to 1000. A process with a score of -1000 is least likely to be killed while a process with a score of 1000 is most likely to be killed. At -1000, a process can use 100% of memory and still avoid being terminated by the OOM killer; however, SR Linux kills the process more frequently.

The **cgroup** parameter and the **oom-score-adj** parameter are shown in the output below.

```
bgp_mgr:
   path: @SRLINUX_BINARY_INSTALL_PREFIX@
   launch-command: @YAML_LAUNCH_ENVIRONMENT@ ./sr_bgp_mgr
   search-command: ./sr_bgp_mgr
   oom-score-adj: 0
   wait-for-config: Yes
   author: 'Nokia'
   restricted-operations: ['reload']
   cgroup: workload.slice/primary
   yang-modules:
       names:
           - "srl_nokia-bgp"
           - "srl_nokia-bgp-vpn"
           - "srl_nokia-rib-bgp"
           - "srl_nokia-system-network-instance-bgp-vpn"
       tools:
           - "srl_nokia-tools-bgp"
       source-directories:
           - "@SRLINUX_FILE_INSTALL_PREFIX@/models/ietf"
           - "@SRLINUX_FILE_INSTALL_PREFIX@/models/srl_nokia/models/common"
           - "@SRLINUX_FILE_INSTALL_PREFIX@/models/srl_nokia/models/interfaces"
           - "@SRLINUX_FILE_INSTALL_PREFIX@/models/srl_nokia/models/network-
instance"
           - "@SRLINUX_FILE_INSTALL_PREFIX@/models/srl_nokia/models/routing-policy"
           - "@SRLINUX_FILE_INSTALL_PREFIX@/models/srl_nokia/models/system"
```

15.8.5 Debugging cgroups

Cgroup debugging capability is available through:

- SR Linux CLI commands
- Linux-provided CLI commands

15.8.5.1 SR Linux cgroup debugging commands

The SR Linux provides CLI commands to perform the following:

- check the usage of existing cgroups
- show information about the OOM adjust score of applications managed by the app_mgr
- show information about the cgroups that are associated with the applications that are managed by the app_mgr
- list all of the applications associated with a specified cgroup

15.8.5.1.1 Checking existing cgroup usage

The output below is an example of checking existing cgroup usage.

```
--{ running }--[  ]--
A:rifa# info from state platform control A cgroup *
    platform {
        control A {
            cgroup /mnt/cgroup/unified/user.slice/default {
                memory-statistics {
```

```
                    current 0
                    current-swap 0
                    anon 0
                    kernel-stack 0
                    slab 0
                    sock 0
                    anon-thp 0
                    file 0
                    file-writeback 0
                    file-dirty 0
                }
                cpuacct-statistics {
                    user 0
                    system 0
                }
            }
            cgroup /mnt/cgroup/unified/workload.slice/primary {
                memory-statistics {
                    current 1621110784
                    current-swap 0
                    anon 1327427584
                    kernel-stack 2396160
                    slab 12939264
                    sock 151552
                    anon-thp 855638016
                    file 250916864
                    file-writeback 0
                    file-dirty 0
                }
                cpuacct-statistics {
                    user 22082989551
                    system 11252275018
                }
            }
            cgroup /mnt/cgroup/unified/workload.slice/secondary {
                memory-statistics {
                    current 7827456
                    current-swap 0
                    anon 4501504
                    kernel-stack 73728
                    slab 2453504
                    sock 8192
                    anon-thp 0
                    file 0
                    file-writeback 0
                    file-dirty 0
                }
                cpuacct-statistics {
                    user 34323091
                    system 12342430
                }
            }
        }
    }
```

15.8.5.1.2 Showing current OOM adjust scores

The output below is an example of showing information about the current OOM adjust scores for all applications that are managed by the app_mgr.

```
--{ running }--[  ]--
A:rifa# info from state system app-management application * oom-score-adj
```

```
    system {
        app-management {
            application aaa_mgr {
                oom-score-adj 0
            }
            application acl_mgr {
                oom-score-adj 0
            }
            application app_mgr {
                oom-score-adj -998
            }
            application arp_nd_mgr {
                oom-score-adj -200
            }
|
|
|
            application vxlan_mgr {
                oom-score-adj 0
            }
            application xdp_lc_1 {
                oom-score-adj -200
            }
        }
    }
```

15.8.5.1.3 Showing cgroup information

The output below is an example of showing information about cgroups that are associated with applications managed by the app_mgr.

```
--{ running }--[  ]--
# info from state system app-management application * cgroup
    system {
        app-management {
            application aaa_mgr {
                cgroup /mnt/cgroup/unified/workload.slice/primary
            }
            application acl_mgr {
                cgroup /mnt/cgroup/unified/workload.slice/primary
            }
            application arp_nd_mgr {
                cgroup /mnt/cgroup/unified/workload.slice/primary
            }
            application bgp_mgr {
                cgroup /mnt/cgroup/unified/workload.slice/primary
            }
|
|
|
            application sshd-mgmt {
                cgroup /mnt/cgroup/unified/workload.slice/secondary
            }
            application supportd {
                cgroup /mnt/cgroup/unified/workload.slice/primary
            }
            application vxlan_mgr {
                cgroup /mnt/cgroup/unified/workload.slice/primary
            }
            application xdp_lc_1 {
                cgroup /mnt/cgroup/unified/workload.slice/primary
            }
```

```
        }
    }
```

15.8.5.1.4 Listing all the applications associated with a specified cgroup

Use the **tools system cgroup command pgrep cgroup** *cgroupname* command to list all the applications associated with a specified group; the output below shows an example.

```
--{ running }--[  ]--
# tools system cgroup command pgrep cgroup workload.slice/primary
+-------+----------------------+
|  Pid  |       Process        |
+=======+======================+
| 3373  | sr_app_mgr           |
| 3385  | sr_supportd          |
| 3402  | sr_device_mgr        |
| 3460  | sr_idb_server        |
| 3471  | sr_aaa_mgr           |
| 3482  | sr_acl_mgr           |
| 3518  | sr_arp_nd_mgr        |
| 3542  | sr_chassis_mgr       |
| 3560  | sr_dhcp_client_mgr   |
| 3574  | sr_evpn_mgr          |
| 3586  | sr_fib_mgr           |
| 3598  | sr_l2_mac_learn_mgr  |
| 3611  | sr_l2_mac_mgr        |
| 3621  | sr_lag_mgr           |
| 3631  | sr_linux_mgr         |
| 3641  | sr_log_mgr           |
| 3651  | sr_mcid_mgr          |
| 3681  | sr_mgmt_server       |
| 3702  | sr_net_inst_mgr      |
| 3724  | sr_oam_mgr           |
| 3743  | sr_sdk_mgr           |
| 3753  | sr_sflow_sample_mgr  |
| 3772  | sr_xdp_lc_1          |
| 3874  | sudo                 |
| 3895  | sudo                 |
| 3902  | sudo                 |
| 3921  | sudo                 |
| 3930  | sr_qos_mgr           |
| 3953  | sr_gnmi_server       |
| 3979  | sr_json_rpc          |
| 3990  | sr_vxlan_mgr         |
| 16740 | sr_json_wkr          |
| 16742 | python3              |
| 17013 | python3              |
| 18474 | sr_json_wkr          |
| 18478 | python3              |
| 18671 | python3              |
| 20671 | sr_lldp_mgr          |
| 21177 | sr_bgp_mgr           |
| 21190 | sr_plcy_mgr          |
| 12753 | sr_l2_static_mac_mgr |
| 27353 | rsyslogd             |
| 9777  | python               |
+-------+----------------------+
```

15.8.5.2 Linux-provided cgroup debugging commands

The following Linux-provided CLI commands are available for debugging cgroups:

- the **systemd-cgls** command
- the **systemd-cgtop** command

The **systemd-cgls** command dumps the cgroup hierarchy. The output below shows an example of the **systemd-cgls** command.

```
[linuxadmin@srlinux unified]$ systemd-cgls
+-1 /usr/lib/systemd/systemd --switched-root --system --deserialize 22
+-system.slice
  +-rngd.service
  | +-2725 /sbin/rngd -f
  +-systemd-udevd.service
  | +-868 /usr/lib/systemd/systemd-udevd
  +-system-serial\x2dgetty.slice
  | +-serial-getty@ttyS0.service
  |   +-1120 login -- linuxadmin
  |   +-8320 -bash
  |   +-9262 sendacct "systemd-cgls"
  |   +-9263 systemd-cgls
  |   +-9264 less
  +-srlinux.service
  | +- 3092 /usr/bin/sudo /opt/srlinux/bin/sr_linux
  | +- 3120 runuser --user srlinux --group srlinux -- /opt/srlinux/bin/sr_linux
  | +- 4019 /bin/bash /opt/srlinux/bin/sr_linux --user-env-switched
  | +- 4039 ./sr_app_mgr
  | +- 4051 ./sr_supportd --server-mode
  | +- 4068 ./sr_device_mgr
  | +- 4219 ./sr_idb_server
  | +- 4229 ./sr_eth_switch
  | +- 4288 ./sr_aaa_mgr
  | +- 4299 ./sr_acl_mgr
  | +- 4314 ./sr_arp_nd_mgr
  | +- 4324 ./sr_chassis_mgr
  | +- 4337 ./sr_dhcp_client_mgr
  | +- 4365 ./sr_evpn_mgr
  | +- 4381 ./sr_fib_mgr
  | +- 4395 ./sr_l2_mac_learn_mgr
  | +- 4411 ./sr_l2_mac_mgr
  | +- 4423 ./sr_lag_mgr
  | +- 4445 ./sr_linux_mgr
  | +- 4494 ./sr_log_mgr
  | +- 4510 ./ntpd -c /etc/ntps.conf -g
  | +- 4526 ./sr_mcid_mgr
  | +- 4574 ./sr_mgmt_server
  | +- 4608 ./sr_net_inst_mgr
  | +- 4626 ./sr_oam_mgr
  | +- 4640 ./sr_sdk_mgr
  | +- 4658 ./sr_sflow_sample_mgr
  | +- 4681 ./sr_xdp_cpm
  | +- 5197 /usr/bin/sudo -Enu root /usr/bin/sudo -Enu gnmirpc bash -c ./sr_gnmi
  | +- 5224 /usr/bin/sudo -Enu root /usr/bin/sudo -Enu jsonrpc bash -c ./sr_json
  | +- 5243 /usr/bin/sudo -Enu gnmirpc bash -c ./sr_gnmi_server
  | +- 5248 ./sr_qos_mgr
  | +- 5261 /usr/bin/sudo -Enu jsonrpc bash -c ./sr_json_rpc
  | +- 5290 ./sr_gnmi_server
  | +- 5302 ./sr_json_rpc
  | +- 5693 /usr/sbin/sshd -f /etc/ssh/sshd_config_mgmt
  | +- 6361 /usr/sbin/rsyslogd -i /var/run/srlinux/rsyslogd.pid
  | +-21936 ./sr_bfd_mgr
  | +-22018 ./sr_bgp_mgr
  | +-22132 ./sr_isis_mgr
```

```
  | +-22242 ./sr_lldp_mgr
  | +-22321 ./sr_mpls_mgr
  | +-22437 ./sr_ospf_mgr
  | +-22588 ./sr_plcy_mgr
  | +-22701 ./sr_static_route_mgr
  | +-22796 sr_json_wkr
  | +-22797 python3 -m srlinux.mgmt.json_rpc.json_rpc_server_main --bind=[::]:40
  | +-22953 ./ntpd -c /etc/mntp.conf -g
  | +-23023 python3 -m srlinux.mgmt.json_rpc.json_rpc_server_main --bind=[::]:40
  | +-23736 sr_json_wkr
  | +-23741 python3 -m srlinux.mgmt.json_rpc.json_rpc_server_main --bind=[::]:40
  | +-24009 python3 -m srlinux.mgmt.json_rpc.json_rpc_server_main --bind=[::]:40
  | +-27488 ./dnsmasq --conf-file=/etc/dnsmasq.conf
  +-polkit.service
  | +-2701 /usr/lib/polkit-1/polkitd --no-debug
  +-ztpapi.service
  | +-1114 /opt/srlinux/ztp/virtual-env/bin/python -m ztp.ztphttp
  +-systemd-journald.service
  | +-840 /usr/lib/systemd/systemd-journald
  +-sshd.service
  | +-985 /usr/sbin/sshd -D
  +-crond.service
  | +-1133 /usr/sbin/crond -n
  +-sr_watchdog.service
  | +-1155 sr_wd
  +-dbus.service
  | +-991 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile
  +-systemd-logind.service
    +-1062 /usr/lib/systemd/systemd-logind
```

The **systemd-cgtop** command dumps the current usage of each cgroup. The output below shows an example of the **systemd-cgtop** command.

```
Path                                    Tasks   %CPU   Memory  Input/s Output/s

/                                         186      -        -        -        -
/system.slice/crond.service                 1      -        -        -        -
/system.slice/dbus.service                  1      -        -        -        -
/system.slice/polkit.service                1      -        -        -        -
/system.slice/rngd.service                  1      -        -        -        -
/system.slice/sr_watchdog.service           1      -        -        -        -
/system.slice/srlinux.service              53      -        -        -        -
/system.slice/sshd.service                  1      -        -        -        -
/system.slic...ial-getty@ttyS0.service      3      -        -        -        -
/system.slice/systemd-journald.service      1      -        -        -        -
/system.slice/systemd-logind.service        1      -        -        -        -
/system.slice/systemd-udevd.service         1      -        -        -        -
/system.slice/ztpapi.service                1      -        -        -        -
```

16 Mirroring

Mirroring copies IPv4 and IPv6 packets seen on a specified source, such as an interface (port) or subinterface (VLAN). and sends the packets to a specific destination, such as a locally attached traffic analyzer or a tunnel toward a remote destination.

By default, the mirrored packets include IPv4/IPv6 headers, as well as Ethernet headers. Traffic from multiple sources can be mirrored to a single destination, although traffic from a specific source cannot be mirrored to multiple destinations.

16.1 Mirror sources

A mirror source can be an interface, including all subinterfaces within that interface. The source can be a single interface (for example, `interface ethernet-1/1`) or a LAG (for example, `interface lag1`). Either a LAG member or LAG port can be mirrored. When a LAG port is configured as a mirror source, mirroring is enabled on all ports making up the LAG.

The source can be a specific VLAN; that is, a subinterface within an interface where VLAN tagging is enabled (for example, `interface ethernet-1/1.1` or `lag1.1`).

You can configure mirroring for traffic in a specific direction (ingress only, egress only) or bidirectional traffic (both ingress and egress).

16.2 Mirror destinations

Traffic from the mirror source can be copied to a local destination (local mirroring). In a local mirroring configuration, both the mirror source and mirror destination reside on the same SR Linux node, as shown in Figure 7: Local mirroring.

In this configuration, the local destination is a Switched Port Analyzer (SPAN).

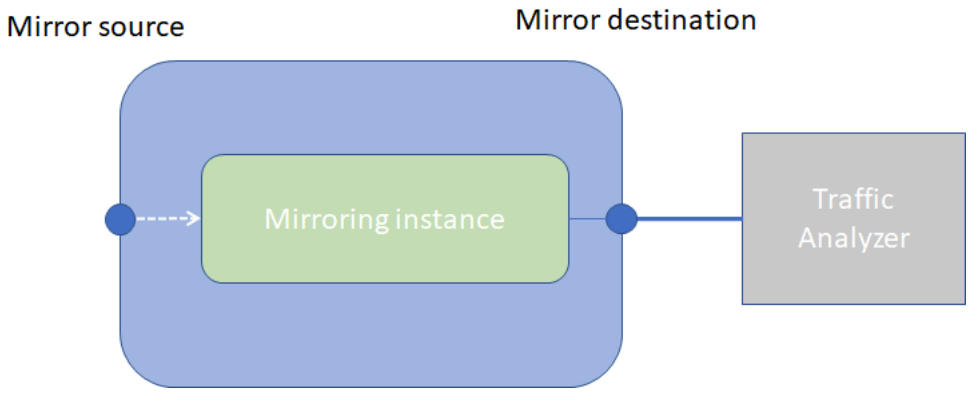

Figure 7: Local mirroring

16.3 Configuring mirroring

To configure mirroring, you configure a mirroring-instance, which specifies the source and destination for the mirrored traffic. Multiple mirror sources can have a single destination, although traffic from a specific source cannot be mirrored to multiple destinations. Only one mirror destination can be configured per mirroring-instance. A mirror destination cannot be reused in multiple mirroring instances.

Within a mirroring-instance, if an interface is configured as mirror source, a subinterface within that interface cannot be added as another mirror source. If a LAG is defined as mirror destination, only the first 8 members of the LAG carry mirrored traffic.

Mirrored traffic is considered Best Effort (BE) Forwarding Class.

16.3.1 Configuring mirroring sources

To configure mirroring, you specify the source and destination for mirrored traffic within a mirroring-instance. The source in a mirroring-instance can be traffic on a specified interface, subinterface, or LAG.

Example: interface source

The following example shows a mirroring-instance configuration with an interface as the source for mirrored traffic:

```
--{ * candidate shared default }--[  ]--
# info system mirroring
    system {
        mirroring {
            mirroring-instance 1 {
                admin-state enable
                mirror-source {
                    interface ethernet-1/5 {
                        direction ingress-egress
                    }
                }
            }
        }
    }
```

16.3.2 Configuring mirroring destinations

In a mirroring-instance, you specify the destination for the mirrored traffic. The mirroring destination can be a local destination residing on the same SR Linux node as the mirroring source.

Example: Local destination

The following enables a subinterface to be a local mirror destination:

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/4 subinterface 1
    interface ethernet-1/4 {
        subinterface 1 {
            type local-mirror-dest
            admin-state enable
            local-mirror-destination {
                admin-state enable
            }
        }
```

```
    }
```

The following configures a mirroring-instance where traffic from the mirror source is mirrored to the subinterface enabled as a local mirror destination:

```
--{ * candidate shared default }--[  ]--
# info system mirroring
    system {
        mirroring {
            mirroring-instance 1 {
                admin-state enable
                mirror-source {
                    interface ethernet-2/1 {
                        direction ingress-egress
                    }
                }
                mirror-destination {
                    local ethernet-1/4.1
                }
            }
        }
    }
```

16.4 Displaying mirroring information

Use the **info from state** command to display mirroring configuration information.

Example:

```
--{ * candidate shared default }--[  ]--
# info from state system mirroring mirroring-instance 2
    system {
        mirroring {
            mirroring-instance 2 {
                admin-state enable
                oper-state down
                oper-down-reason local-mirror-subif-down
                mirror-source {
                    interface lag1 {
                        direction ingress-egress
                    }
                }
                mirror-destination {
                    local lag25.1
                }
            }
        }
    }
```

16.5 Displaying mirroring statistics

You can use the **info from state** command to display the outgoing mirrored packets/octets per interface.

Example:

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

```
# info from state interface ethernet-1/1 statistics | grep mirror
            out-mirror-octets 0
            out-mirror-packets 0
```

17 UFT profiles

The processors in 7220 IXR-D1, 7220 IXR-D2/D3, and 7220-IXR H2/H3 systems use shared memory tables known as Unified Forwarding Tables (UFTs). With UFTs, the processor has a set of shared banks that can be partitioned to support the following types of forwarding table entries:

- Learned MAC addresses
- IP host entries
- IP longest-prefix-match routes

The default settings for the shared bank allocations can be changed by modifying the UFT profile for the system.

17.1 Shared bank partitioning for SR Linux systems

The number of shared banks, the size of each shared bank, and the way they can be divided depend on the SR Linux system, as described in Table 17: Shared bank configuration for SR Linux systems.

Table 17: Shared bank configuration for SR Linux systems

SR Linux system	Shared bank configuration
7220 IXR-D1	6 shared banks 16K single-wide/SW or 8K double-wide/DW entries per shared bank • 1 SW entry can store 1 IPv4 host entry or 1 MAC address. • 1 DW entry can store 1 IPv6 host entry. • If a bank has any DW entry, it is forced into DW mode. If ALPM is enabled, it requires 4 shared banks; each of the remaining banks can be allocated to either IP host entries or MAC addresses.
7220 IXR-D2/D3	8 shared banks 32K single-wide of 16K double-wide entries per shared bank • 1 SW entry can store 1 IPv4 host entry or 1 MAC address. • 1 DW entry can store 1 IPv6 host entry. • If a bank has any DW entry, it is forced into DW mode. ALPM can be enabled in 8-bank (high-scale) mode; each of the remaining banks, if there are any, can be allocated to either IP host entries or MAC addresses.
7220 IXR-H2/H3	8 shared banks 8K entries per shared bank

SR Linux system	Shared bank configuration
	ALPM is enabled by default. ALPM uses all of the shared banks.

17.2 LPM table partitioning

IP FIB scale depends significantly on the number of UFT banks used for ALPM, but also depends on the partitioning of the hardware LPM table, which is an always-present TCAM + SRAM table that stores IP LPM route entries.

When ALPM is not active, IP FIB scale is entirely determined by the size and partitioning of this table. When ALPM is active, the hardware LPM table is used as a first-search table into the ALPM banks, so it also plays an important role.

If the **ipv6-128bit-lpm-entries** parameter is configured to be greater than zero, the hardware LPM table is partitioned into two sub-blocks: a single-wide sub-block and a double-wide sub-block:

- The single-wide sub-block can store IPv4 routes (each consuming half of a single-wide entry) and IPv6 routes up to /64 prefix length (each consuming a single-wide entry).
- The size of the double-wide sub-block is controlled by the **ipv6-128bit-lpm-entries** parameter, and this sub-block can store IPv6 routes up to /128 prefix length (each consuming a double-wide entry).

  It can also store IPv4 routes and IPv6 routes up to /64 prefix length; inside the double-wide sub-block, IPv4 routes consume half of a single-wide entry, and IPv6 routes up to /64 prefix length require a double-wide entry.

17.3 Default UFT allocations for SR Linux systems

The default UFT shared bank allocations and hardware LPM table parameters for each SR Linux system are summarized in Table 18: Default UFT allocations and hardware LPM table parameters.

Table 18: Default UFT allocations and hardware LPM table parameters

SR Linux system	Default UFT allocations and L3DEFIP table parameters
7220 IXR-D1	Extra IP host shared banks: 3 Extra MAC address shared banks: 3 ALPM: disabled **ipv6-128bit-lpm-entries**: 1024
7220 IXR-D2/D3	Extra IP host shared banks: 4 Extra MAC address shared banks: 4 ALPM: disabled **ipv6-128bit-lpm-entries**: 2048
7220 IXR-H2/H3	ALPM: enabled **ipv6-128bit-lpm-entries**: 512

17.4 Configuring a UFT profile

You can change the settings for the UFT shared bank allocations and hardware LPM table parameters from the defaults listed in Table 18: Default UFT allocations and hardware LPM table parameters by modifying the system UFT profile.

To change the settings for the UFT shared bank allocations and hardware LPM table parameters from the defaults, modify the system UFT profile as shown in the following example.

Example:

This example configures a UFT profile for a 7220 IXR-D1 system. The UFT profile enables ALPM and configures 32K extra IP host entries from the UFT shared banks.

```
--{ * candidate shared default }--[  ]--
# info platform resource-management unified-forwarding-resources
    platform {
        resource-management {
            unified-forwarding-resources {
                alpm enabled
                requested-extra-ip-host-entries 32768
                ipv6-128bit-lpm-entries 1024
            }
        }
    }
```

Note:

UFT profile configuration changes do not take effect immediately when the changes are committed; they take effect the next time XDP is restarted.

17.5 Displaying UFT profile information

Use the **info from state** command to display the UFT profile, including the extra number of host entries and MAC address entries allocated from UFT shared banks.

Example:

```
--{ * candidate shared default }--[  ]--
# info from state platform resource-management unified-forwarding-resources
    platform {
        resource-management {
            unified-forwarding-resources {
                xdp-restart-required true
                alpm enabled
                requested-extra-ip-host-entries 32768
                allocated-extra-ip-host-entries 32768
                allocated-extra-mac-entries 0
                ipv6-128bit-lpm-entries 1024
            }
        }
    }
```

In the example, the `xdp-restart-required` leaf is shown as `true` if a change has been committed to one or more of the configurable values in the `unified-forwarding-resources` container, but XDP

has not yet been restarted. Until XDP is restarted, the operational values are still the values initialized at the last XDP restart.

18 Maintenance Mode

SR Linux maintenance mode allows you to take a network element out of service so that maintenance actions can be performed; for example, to upgrade the software image on a router.

Using SR Linux maintenance mode, you can do this with minimal impact on traffic. SR Linux maintenance mode works as follows:

1. A maintenance group is configured that specifies the resources to be taken out of service. See Configuring a maintenance group.
2. A maintenance profile is configured that specifies policy changes to apply when the group is in maintenance mode. A maintenance profile is associated with each maintenance group. See Configuring a maintenance profile.

   The usual intent of the policy changes is to de-preference paths through the maintenance group so that traffic is diverted elsewhere.
3. The maintenance group is placed into maintenance mode, which applies the policies in the associated maintenance profile. See Placing a maintenance group into maintenance mode.
4. The user monitors the traffic on the interfaces in the maintenance group. When the traffic rate falls below a threshold, the user shuts down the members of the maintenance group and performs the required service. See Taking a maintenance group out of service.
5. After the service is completed, the user takes the maintenance group out of maintenance mode, which disables the policies in the associated maintenance profile, and restores traffic on the original paths. See Restoring the maintenance group to service.

18.1 Configuring a maintenance group

A maintenance group specifies a set of network resources to be taken out of service when maintenance mode is enabled. For example, a maintenance group can consist of one or more BGP neighbors or peer groups belonging to a one or more network instances.

Specify a set of network resources in a maintenance group. The network resources in the maintenance group are taken out of service when maintenance mode is enabled.

Example:

The following example configures maintenance group mgroup1, consisting of the BGP neighbors in peer group headquarters1, which exists in the default network instance, as well as BGP neighbors in peer group headquarters2, which exists in network instance "black". In the example, maintenance group mgroup1 is associated with maintenance profile mprof1.

```
--{ candidate shared default }--[  ]--
# info system maintenance
    system {
        maintenance {
            group mgroup1 {
                maintenance-profile mprof1
                maintenance-mode {
                    admin-state disable
                }
```

```
                members {
                    bgp {
                        network-instance default {
                            peer-group [
                                headquarters1
                            ]
                        network-instance black {
                            peer-group [
                                headquarters2
                            ]
                        }
                    }
                }
            }
        }
    }
```

18.2 Configuring a maintenance profile

A maintenance profile specifies policy changes that are applied to members of a maintenance group when maintenance mode is enabled for the group.

The following example configures a routing policy and a maintenance profile that references the routing policy.

Example:

The following example defines the routing policy drain-with-as-path-prepend:

```
--{ candidate shared default }--[  ]--
# info routing-policy policy drain-with-as-path-prepend
    routing-policy {
        policy drain-with-as-path-prepend {
            default-action {
                accept {
                    bgp {
                        as-path {
                            prepend {
                                as-number auto
                                repeat-n 3
                            }
                        }
                    }
                }
            }
        }
    }
```

The following example defines a maintenance profile that references the routing policy drain-with-as-path-prepend.

```
--{ candidate shared default }--[  ]--
# info system maintenance
    system {
        maintenance {
            profile mprof1 {
                bgp {
                    import-policy drain-with-as-path-prepend
                    export-policy drain-with-as-path-prepend
                }
```

```
            }
        }
    }
```

18.3 Placing a maintenance group into maintenance mode

When a maintenance group is placed into maintenance mode, it applies the policies in the associated maintenance profile to the resources in the maintenance group.

To place a maintenance group into maintenance mode, change the setting for **maintenance-mode** in the group configuration to **enable**, then commit the configuration.

Example:

The following example enables maintenance mode for maintenance group mgroup1, and commits the configuration. The policies configured in the maintenance profile associated with mgroup1 are applied to the resources configured in mgroup1.

```
--{ candidate shared default }--[  ]--
# info system maintenance group mgroup1
    system {
        maintenance {
            group mgroup1 {
                maintenance-mode {
                    admin-state enable
                }
            }
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# commit stay
All changes have been committed. Starting new transaction.
```

18.4 Taking a maintenance group out of service

After enabling maintenance mode for a maintenance group, monitor the traffic on the interfaces in the group. When the traffic rate drops to an acceptable level, shut down the members of the maintenance group and perform the required service.

Monitor traffic for an interface using the **info from state** command to show interface traffic statistics. When the traffic rate reaches a low-enough level, administratively disable the interface.

Example:

To monitor an interface:

```
--{ running }--[  ]--
# info from state interface ethernet-1/2 statistics
    interface ethernet-1/2 {
        statistics {
            in-octets 46969
            in-unicast-pkts 492
            in-broadcast-pkts 0
            in-multicast-pkts 34
```

```
            in-discards 0
            in-errors 0
            in-unknown-protos 0
            in-fcs-errors 0
            out-octets 46169
            out-unicast-pkts 490
            out-broadcast-pkts 1
            out-multicast-pkts 25
            out-discards 0
            out-errors 0
            carrier-transitions 1
```

To shut down the interface:

```
--{ * candidate shared default }--[  ]--
# interface ethernet-1/2 admin-state disable
# commit stay
All changes have been committed. Starting new transaction.
```

18.5 Restoring the maintenance group to service

After performing the required maintenance operations, restore the maintenance group to service.

To restore the maintenance group to service, change the setting for the `maintenance-mode` state in the group configuration to `disable`, then commit the configuration.

Example:

The following example takes maintenance group mgroup1 out of maintenance mode:

```
--{ candidate shared default }--[  ]--
# info system maintenance group mgroup1
    system {
        maintenance {
            group mgroup1 {
                maintenance-mode {
                    admin-state disable
                }
            }
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# commit stay
All changes have been committed. Starting new transaction.
```

Customer document and product support

Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback