Nokia Service Router Linux
7215 Interconnect System
7220 Interconnect Router
7250 Interconnect Router
7730 Service Interconnect Router

Release 26.3

Configuration Basics Guide

3HE 22246 AAAA TQZZA
Edition: 01
March 2026

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 Software Release Notes* for information on features supported in each load.

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

1.1 Precautionary and information messages

The following are information symbols used in the documentation.

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

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

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

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

1.2 Conventions

The 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.
- 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.

The following table outlines platform grouping conventions used in the SR Linux documentation suite.

Note: Some platforms in the 7250 IXR support mixed systems. For more information about mixed system support, see "Chassis types" in the *Configuration Basics Guide*.

Table 1: Platform grouping legend

Platform group	Description
7215 IXS	7215 IXS-A1
7220 IXR	All 7220 IXR platforms
7220 IXR-D*x*	7220 IXR-D1, 7220 IXR-D2, 7220 IXR-D2L, 7220 IXR-D3, 7220 IXR-D3L, 7220 IXR-D4, 7220 IXR-D5
7220 IXR-H*x*	7220 IXR-H2, 7220 IXR-H3, 7220 IXR-H4, 7220 IXR-H4-32D, 7220 IXR-H5-32D, 7220 IXR-H5-64D, 7220 IXR-H5-64O
7250 IXR[1]	7250 IXR platforms
7250 IXR Gen 2	7250 IXR-6, 7250 IXR-10
7250 IXR Gen 2c+	7250 IXR-6e with IMM2, 7250 IXR-10e with IMM2, 7250 IXR-X1b, 7250 IXR-X3b
7250 IXR Gen 3	7250 IXR-6e with IMM3, 7250 IXR-10e with IMM3, 7250 IXR-18e, 7250 IXR-X4
7250 IXR-6e/10e (mixed system)	7250 IXR-6e (mixed system)[2], 7250 IXR-10e (mixed system)[2]
7730 SXR	7730 SXR-1-32D, 7730 SXR-1d-32D, 7730 SXR-1x-44S

1 References to the 7250 IXR platform group may be appended with (including mixed systems) or (excluding mixed systems) to indicate mixed system support.

2 References to this platform as part of 7250 IXR (mixed system) indicate mixed system support of 7250 IXR Gen 2c+ (IMM2) and 7250 IXR Gen 3 (IMM3). That is, the 7250 IXR-6e and 7250 IXR-10e can hold and support both IMM2 and IMM3 at the same time.

2 What's new

This section lists the changes that were made in this release.

Table 2: What's new in Release 26.3.1

Topic	Location
Mixed chassis mode	Chassis types
Supports 7250 IXR Gen 2c+ and 7250 IXR (including mixed)	Non-Stop Routing
SSH port forwarding	Configuring SSH port forwarding
NTP TLS support	NTP TLS support
Enhanced password complexity rules	Password security for local users
gRPC tunnel support	gRPC tunnel
Updated platform support for VRRP	Supported platforms
DNS instances bound to specific network-instances	DNS instances
New OpenConfig-based model for configuring next-hop-groups	Static routes
GUE tunnel encapsulation using static routes	New model for configuring static routes
Route leaking based on BGP attributes	Matching BGP attributes in a route-leaking-export policy

3 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 agent.

3.1 System information

The following sections describe the system information components.

Contact

Use the **system information contact** command to specify the name of a system administrator, IT staff member, or other administrative entity.

The following shows an example configuration of the **system information contact** command.

Example: Contact command configuration

```
--{ * candidate shared default }--[  ]--
A:root@srl1# info with-context system information
    system {
        information {
            contact user1
        }
    }
```

Coordinates

You can use the **coordinates** command to configure the GPS location of the device. In the **system information coordinates** command context, set the following parameters to configure the system coordinates:

- **height** — height in meters
- **latitude** — decimal value between -90 and 90, to a maximum of 6 decimal places
- **longitude** — decimal value between -180 and 180, to a maximum of 6 decimal places

The coordinates are based on the World Geodetic System 1984 (WGS84) accurate to 6 decimal places.

The height value is the Height Above Ellipsoid (HAE) which is the height in meters above the Earth's surface reported by a GNSS sensor. This value is not the same as height above mean sea-level (MSL).

The following shows an example configuration of the **system information coordinates** command.

Example: Coordinates command configuration

```
--{ * candidate shared default }--[  ]--
A:root@srl1# info with-context system information
    system {
        information {
```

```
            coordinates {
                latitude 50.000000
                longitude 50.000000
                height 10
            }
        }
    }
```

Location

Use the **system information location** command to specify the physical system location of the device (for example, the city, building address, floor, room number, and so on).

The following shows an example configuration of the **system information location** command.

Example: Location command configuration

```
--{ * candidate shared default }--[  ]--
A:root@srl1# info with-context system information
    system {
        information {
            location address1
        }
    }
```

Protobuf-metadata

Use the **system information protobuf-metadata** command to provide contextual metadata in a binary format using Protocol Buffers (protobuf).

The following example shows the configuration of the protobuf-metadata command.

Example: Protobuf-metadata command configuration

```
--{ * candidate shared default }--[  ]--
A:root@srl1# info with-context system information
    system {
        information {
            protobuf-metadata MTA=
        }
    }
```

3.2 Chassis types

Each 7250 IXR-6e/10e chassis operates in one of three modes (or types) and this determines the supported functionality and the set of Integrated Media Modules (IMMs) and Switch Fabric Modules (SFMs) that can become operational in the chassis. The chassis type is defined in terms of the ASIC generation used on the IMMs.

Table 3: Chassis types

Chassis	ASIC type	Chassis type
7250 IXR-6e	Gen 2c+	7250 IXR-6e-gen2cp
7250 IXR-6e	Gen 3	7250 IXR-6e-gen3

Chassis	ASIC type	Chassis type
7250 IXR-6e	mixed Gen 2c+ and Gen 3	7250 IXR-6e-gen2cp-3
7250 IXR-10e	Gen 2c+	7250 IXR-10e-gen2cp
7250 IXR-10e	Gen 3	7250 IXR-10e-gen3
7250 IXR-10e	mixed Gen 2c+ and Gen 3	7250 IXR-10e-gen2cp-3

The chassis type definition ensures that all installed cards have compatible features and scale characteristics.

The primary decision for the operator is which chassis mode to operate in, and this selection guides the subsequent installation of IMMs and SFMs.

Note: Operators can configure the intended chassis type during initial commissioning and may change it later when upgrading or replacing IMMs and SFMs. For platform specific IMM/SFM details, see the SR Linux Software Release Notes.

- gen2cp-only: When a 7250 IXR-6e/10e chassis boots up as a **gen2cp-only** chassis, the SFMs and IMMs that are incompatible (7250 IXR Gen 3 IMMs and SFMs) are prevented from booting and remain operationally down.
- gen3-only: When a 7250 IXR-6e/10e chassis boots up as a **gen3-only** chassis, the SFMs and IMMs that are incompatible (7250 IXR Gen 2c+ IMMs and SFMs) are prevented from booting and remain operationally down.
- gen2cp-gen3-mixed: When a 7250 IXR-6e/10e chassis boots up as a **gen2cp-gen3-mixed** chassis, the SFMs that are incompatible (7250 IXR Gen 2c+ SFMs) are prevented from booting and remain operationally down.

WARNING: The **gen2cp-gen3-mixed** mode offers limited functionality compared to **gen2cp-only** and **gen3-only** modes and is not recommended for general-purpose deployments. Refer to the SR Linux Software Release Notes for details on the mixed-mode restriction.

3.2.1 Configuring chassis type

Procedure

On a 7250 IXR-6e/10e platform, use the **tools.platform.chassis.type** command to configure the chassis type (referred to as `preferred` type).

The parameter **type** can be set to **gen2cp-only**, **gen3-only** or **gen2cp-gen3-mixed**. Run the command **tools.platform.chassis.reboot** to apply the configuration.

Example: Display current running chassis type using info from state command

```
--{ + candidate shared default }--[  ]--
# info from state with-context platform chassis type
    platform {
        chassis {
            type "7250 IXR-10e-gen2cp-3"
        }
    }
```

Example: Display current running chassis type using show platform command

```
--{ running }--[  ]--
# show platform chassis
------------------------------------------------------------
Type             : 7250 IXR-10e-gen2cp-3
Last Boot type   : normal
HW MAC address   : 1A:26:00:FF:00:00
Slots            : 8
Oper Status      : up
Last booted      : 2026-01-22T18:31:21.153Z
Last change      : 2026-01-22T18:31:21.153Z
Part number      : Sim Part No.
CLEI code        : Sim CLEI
Serial number    : Sim Serial No.
Manufactured date: 2019-01-01T00:00:00Z
------------------------------------------------------
```

Example: Display configured chassis type using info from state command

The following configuration example shows that the chassis is currently operating as a **gen2cp-only** type and has been configured, using the **tools.platform.chassis.type** command, to run as a **gen3-only** type. The parameter **chassis-reboot-required** is set to `true` indicating that a chassis reboot is necessary to complete the transition from mixed Gen 2c+ to Gen 3. You must execute the **tools.platform.chassis.reboot** command for the chassis type change to take effect.

```
--{ + candidate shared default }--[  ]--
# info from state with-context platform chassis mode
    platform {
        chassis {
            mode {
                running gen2cp-only
                configured gen3-only
                chassis-reboot-required true
            }
        }
    }
```

After you reboot, the command output is as follows:

```
--{ + candidate shared default }--[  ]--
# info from state with-context platform chassis mode
    platform {
        chassis {
            mode {
                running gen3-only
                configured gen3-only
                chassis-reboot-required false
            }
        }
    }
```

3.2.2 Selecting chassis type at boot

When the active CPM starts, it determines whether a chassis type has been previously selected or configured. The following describes the behavior depending on whether a chassis type has been previously selected or configured.

- No previously selected/configured chassis type:
  - The CPM initially assumes the chassis type is **gen2cp-only** (referred to as the `estimated` type) and waits for at least one IMM to boot.
  - If the first IMM detected by the CPM is a 7250 IXR Gen 2c+ IMM, this confirms the chassis type and selected chassis type as **gen2cp-only** (referred to as `tentative` type). The 7250 IXR Gen 3 IMMs are blocked from booting and remain operationally down.
  - If the first IMM detected by the CPM is a 7250 IXR Gen 3 IMM, this confirms the chassis type and selected chassis type as **gen3-only** (referred to as `tentative` type). The 7250 IXR Gen 2c+ IMMs are blocked from booting and remain operationally down.
- Previously configured chassis type:
  - The CPM applies the previously selected/configured chassis type immediately.
  - If the chassis type specified is **gen2cp-only** (referred to as `preferred` type), the 7250 IXR Gen 3 IMMs and SFMs are prevented from booting and remain operationally down.
  - If the chassis type specified is **gen3-only** (referred to as `preferred` type), the 7250 IXR Gen 2c+ IMMs and SFMs are prevented from booting and remain operationally down.
  - If the chassis type specified is **gen2cp-gen3-mixed** (referred to as `preferred` type), all 7250 IXR Gen3 and Gen2c+ IMMs will boot. However, 7250 IXR Gen 2c+ SFMs are prevented from booting and remain operationally down.

When the standby CPM starts, it assumes the chassis type as **gen2cp-only** (the `estimated` type) and waits to receive the selected chassis type from the active CPM during synchronization.

If the chassis type received from the active CPM differs from the standby CPM's estimated type, the standby CPM updates its configuration accordingly and reboots to apply the change.

3.3 Configuring a hostname

Procedure

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

The hostname must be a unique name on the network, and it 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 with-context system name
    system {
        name {
            host-name 3-node_srlinux-A
        }
    }
```

3.4 Configuring a domain name

Procedure

The SR Linux device uses its hostname, combined with a domain name, to form its 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 with-context system name
    system {
        name {
            host-name 3-node_srlinux-A
            domain-name mv.usa.nokia.com
        }
    }
```

3.5 DNS instances

The SR Linux device uses DNS to forward name resolution requests or for operational commands, such as **ping**.

In SR Linux, you can configure a DNS instance that is bound to a specific network-instance. A DNS instance specifies settings for providing DNS resolution for all applications within the network-instance. A network-instance can be bound to a single DNS instance.

In a DNS instance, you can specify up to three DNS servers for the network-instance 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` performs a DNS lookup for `srlinux.nokia1.com`, and if unsuccessful, performs a DNS lookup for `srlinux.nokia2.com`.

A DNS instance supports configuration of static DNS entries. Static DNS entries allow resolution of hostnames that may not be in the DNS servers used by the network-instance. 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.

Within a DNS instance, you can specify the preferred source address when the SR Linux device acts as a DNS client in a network-instance with multiple possible source addresses.

3.5.1 Configuring a DNS instance

Procedure

To configure a DNS instance, you specify the network-instance to which the DNS instance is bound, a search list of DNS suffixes for resolving single-label names, DNS servers to resolve hostname, and IPv4 and IPv6 static DNS entries for specific hosts. In a DNS instance, you can also specify a preferred source address the SR Linux device uses when acting as a DNS client.

Example: Configure parameters in a DNS instance

In the following example, a DNS instance bound to the mgmt network-instance 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.

```
--{ candidate shared default }--[  ]--
# info with-context system dns-instance d1
    system {
        dns-instance d1 {
            network-instance mgmt
            search-list [
                nokia1.com
                nokia2.com
            ]
            server-list [
                192.0.2.1
                192.0.2.2
            ]
            host-entry srlinux.nokia.com {
                ipv4-address [
                    192.0.2.3
                ]
                ipv6-address [
                    2001:db8:123:456::11:11
                ]
            }
        }
    }
```

Example: Configure the preferred source address for DNS requests

A DNS instance can specify the preferred source IPv4 or IPv6 address for the SR Linux device acting as a DNS client in a network-instance with multiple possible source addresses. The source address must belong to the network-instance where DNS is configured.

The following example configures **source-address** parameter in a DNS instance so that DNS requests are sent to the configured DNS servers using a source address of 192.168.12.1.

```
--{ * candidate shared default }--[  ]--
# info with-context system dns-instance d1
    system {
        dns-instance d1 {
            network-instance mgmt
            source-address 192.168.12.1
            server-list [
                192.0.2.1
                192.0.2.2
            ]
        }
```

```
    }
```

3.6 Configuring the management network-instance

Procedure

The SR Linux device is primarily managed 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 with-context 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
            }
        }
    }
```

3.7 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:

- **Secure Shell (SSH)**

  SSH is a standard method for accessing network devices. See Enabling an SSH server.
- **File Transfer Protocol (FTP)**

  The FTP is a secure method for transferring files to and from network devices. See Configuring FTP.
- **console**

  The console provides access to the SR Linux CLI via direct connection to a serial port on the device.
- **gRPC Network Management Interface (gNMI)**

gNMI is 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 gRPC server.

- **JSON-RPC**

  JSON-RPC is a protocol with the ability to retrieve and set configuration and state using a JSON-RPC API. See JSON-RPC server.

- **Simple Network Management Protocol (SNMP)**

  SNMP is a commonly used network management protocol. The SR Linux device supports SNMPv2 and SNMPv3 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.

3.7.1 Enabling an SSH server

Procedure

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 with-context system ssh-server
    system {
        ssh-server {
            network-instance mgmt {
                admin-state enable
                source-address [
                    192.0.2.1
                    192.0.2.2
                ]
            }
            network-instance default {
                admin-state enable
                source-address [
                    192.0.2.3
                    192.0.2.4
                ]
            }
        }
    }
```

3.7.1.1 Configuring SSH key-based authentication

Procedure

The SR Linux SSH server supports RSA public-private key and SSH certificate authentication, where an SSH client provides either a signed message that has been encrypted by a private key or a valid certificate that has been issued by a trusted authority. If the SSH client's corresponding public key or certificate is configured on the SR Linux, the SSH server can authenticate the client using either RSA public key or SSH certificate method.

The SR Linux attempts to authenticate a user using public-key authentication first, then SSH certificates, and finally password authentication if the previous methods fail.

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 2048 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 with a CLI command.

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

```
--{ candidate shared default }--[  ]--
# info system aaa authentication user srlinux ssh-key
    system {
        aaa {
```

```
            authentication {
                user srlinux {
                    ssh-key [
                        "ssh-rsa  AAAAB3NzaC1yc2EAAAADAQABAAABAQD60yC5yh9ZmCLTKCphf
Ji46NMXAZtjC67LDgYjrMBJqOiQoItR0+vAh/6SnxExr8RFHh6h0zjyY2JSDFBZ40sY9Qd
Uh8yFEL034BX313zvCOaJ8juj+txUnD7DEzMjHiAurmPPF5d1wLJ1Udvur6jGUVae1zgBXxmR7QV/omMEFDdr
Fmbpzt1KBaLSMd8DYSrNHczHcld2asA/6L6XrAfsvyTSD19aw6Xku5RysqRf0/BwAKWB7b5wgmeT0p2TlsEAiU4Bo
ZqUJmstM8SyfH8JX7w2qjh55itngV7KZFrwZI7asyvIQkag3ETaHNkMaEogqsNmwDzCDKDMMjmAlKyP
 srlinux@example.com"
                    ]
                }
            }
        }
    }
```

The value of the **ssh-key** string is the SSH public key, for example, `id_rsa.pub`. If multiple public keys are configured for a user, they are tried in the order they were configured.

3.7.2 SSH host keys

An SSH host key is generated when an SSH server is enabled for a network instance on an SR Linux device. This host key is the public key.

SSH clients store the public host keys of the servers to which they are connected.

A host key that is stored is referred to as a *Known host key*. The files `/etc/ssh/known_hosts` and `.ssh/known_hosts` present in the user's home directory, contain the host public keys for all the known hosts.

In SR Linux, the server-side keys are stored in `/etc/ssh`, and the filenames begin with `ssh_host_[dsa/ecdsa/ed25519/rsa]_key`. For information about host key authentication and preservation, see Host key authentication and preservation.

For information about SSH server host certificate configuration, see Configuring host certificate.

3.7.2.1 Host key authentication and preservation

When an SSH client connects to a server, the server offers its host key as identification. If this is the first time the user has connected to the server, the client prompts the user to accept the host key. After the user accepts the key, the host key and the host-name used to connect to the server gets appended to the list of known hosts. In subsequent connections to the same server, the SSH client expects the server to return the same host key.

If the host key presented by the server on a subsequent connection is different from the one saved on the user's local system, the SSH client refuses to proceed with the connection, and instead, displays the following warning message:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:4xnagTqYdQyLGJl0XhvAvyBcrRL2nZ8vSRXTYfcIYe0.
Please contact your system administrator.
```

```
Add correct host key in /Users/test_user/.ssh/known_hosts to get rid of this message.
Offending ED25519 key in /Users/test_user/.ssh/known_hosts:74
Host key for 10.0.0.1 has changed and you have requested strict checking.
Host key verification failed.
```

The message suggests that the server may be pretending to be the intended server to intercept passwords or there could be a change in the host key.

Without correct verification techniques, a periodic change in an SSH host key is indistinguishable from a man-in-the-middle (MITM) attack. Users may be tempted to accept frequent warning messages without verifying them, which can increase the vulnerability to MITM attacks instead of reducing it.

To overcome this, you can enable the `preserve` option for the SSH server host key. This ensures the SSH server host keys in `/etc/ssh` are preserved and restored after a system reboot or SSH server restart. When the `preserve` option is not enabled, the SSH server host key remains valid only until the node is restarted or the SSH server is stopped and restarted.

Note:
This feature is non-backward compatible in releases before 23.7.

For information about enabling the `preserve` option to prevent SSH host key regeneration upon reboot, see Preserving SSH host key.

3.7.2.1.1 Preserving SSH host key

Procedure

To prevent SSH server host key regeneration upon reboot, set the `preserve` option for SSH under **system ssh-server host-key**.

Example

The following example sets the `preserve` option for SSH:

```
--{ * candidate shared default }--[ ]--
# info with-context system ssh-server host-key
    system {
        ssh-server {
            host-key {
                preserve true
            }
        }
    }
```

- When `preserve` option is set to **true**, the SSH server host keys in `/etc/ssh` are saved and restored after a system reboot or SSH server restart.
- When `preserve` option is set to **false**, the SSH server host keys in `/etc/ssh` are removed and regenerated on each system reboot or SSH server restart .

3.7.3 SSH certificates

The SR Linux SSH server supports RSA public-private key and SSH certificate authentication.

When using public key cryptography for authentication, the public key from every client must be copied to every server that the client intends to log into. This type of authentication method does not scale well and requires significant administrative effort to maintain.

As an alternative, OpenSSH supports the creation of simple SSH certificates and associated CA infrastructure. Using a public key from a certificate authority (CA) to authenticate client certificates removes the need to copy keys between multiple systems.

The SSH certificates are signed with a standard SSH public key using the **ssh-keygen** utility. The utility supports two types of certificates: user and host certificates. User certificates authenticate users to servers, while the host certificates authenticate server hosts to users.

Note:
If a user has both an RSA public key and an SSH certificate set up for authentication, the SSH client tries the public key first, and then the certificate.

The high-level steps for enabling SSH certificate authentication are,

- Create a CA key pair and configure the CA public key (trust-anchors) on the SSH server. See Configuring CA public key (trust-anchors).
- Create a user SSH certificate, sign it with the CA private key and SSH principal, and configure an SSH principal for each user. See Configuring SSH principals.
- Create a host SSH certificate, sign it with the CA private key, and configure the SSH host certificate on the SSH server, see Configuring host certificate.
- Set up CA public key as the `cert-authority` for all client nodes. See Trusting CA public key.
- Configure a revoked public user key list. See Configuring revoked key list.

3.7.3.1 Configuring CA public key (trust-anchors)

Procedure

SR Linux allows the configuration of CA public keys, also known as trust-anchors, to verify SSH certificates. The SSH certificates are used by the `sshd` service to authenticate clients (users) connecting via SSH. The CA signing key pair (CA public and private key) is generated by running the `ssh-keygen` command on a server designated as the CA machine.

Example: Generate CA

The following example generates CA signing key pair.

```
ssh-keygen -f ca
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ca
Your public key has been saved in ca.pub
The key fingerprint is:
SHA256:GjGiO2dJV2tKBcuCjC5skMn8cfJ/iToZ/4oR/F0hhAE user@node
The key's randomart image is:
+---[RSA 3072]----+
|     Eo.o.       |
|o= . . +.        |
|=oooo.= o. .     |
|+ ..*o = .. .    |
```

```
|.+...++ S  .     |
|o  o =+=o o      |
|  o +.*+ +       |
|   + o+..        |
|     o..o.       |
+----[SHA256]-----+
```

Example: Configure trust-anchors

The following example configures the `trust-anchors` that verifies the SSH certificates. The `trust-anchors` configuration is a global-setting and is applicable to all SSH servers.

```
--{ candidate shared default }--[  ]--
# info with-context system ssh-server trust-anchors
    system {
        ssh-server {
            trust-anchors [
                "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCujqWgz360aQ6y+7pp3oKjofkbQFvb
Pjr9g0jV2xyOCBjidFH8Czr9QaasfFRIHDFxeydNu/Jt2HtQhLekrUQ8oKjArytR72pLreMzc7tITGDdQuGNZDVNwEefeCbj
XrJ2LPi+fFwvZBoFM7BNYn/Hx4HEHeGyFKGZ5RStx/bMiMW2xVwyfhR/s5wKqXV/kUsxQlyWJEKdjLhRzBbDqDNr3alOJtg
EhOQcD6kG7nCMKdXspjie2Tjha1khs7Ecg/ff8t/Qgf43H4E/imP0v1HJHsiqBgtZxK3wE7TiGDNY8d2LQJblNspzh
Le4QOdqwficozk9QkIodQawWMgcf3UXiKz/akH0ZJVjIc7ETsbC9PgxM0JmXgHVkakq37ExPLkGXV2dEQ3lJhkcC8Kqa
+guHqyxF7ETw4SOnDxKfdROfUnouYgVLDpwhIHXRDzvBOPtu5OH9iK0adf2ceQu7wcQAmGuBzWAWKqZlNKb9JfNCp
AlvdTGpxGEGBmXEl46r3E= user@node"
            ]
        }
    }
```

3.7.3.2 Configuring SSH principals

Procedure

SR Linux allows configuration of SSH principals for users.

A client machine generates a user certificate using the `ssh-keygen` tool and signs it with the CA certificate and SSH principal value.

Example

The following is an example of a signed user certificate (`srl-user1-cert.pub`) with SSH principals.

```
--{ candidate shared default }--[  ]--
# ssh-keygen -L -f srl-user1-cert.pub
srl-user1-cert.pub:
        Type: ssh-rsa-cert-v01@openssh.com user certificate
        Public key: RSA-CERT SHA256:LBPq+l+ksZi++61XoVFGch5Sf3xrksQ5mY70TR3iai4
        Signing CA: RSA SHA256:GjGiO2dJV2tKBcuCjC5skMn8cfJ/iToZ/4oR/F0hhAE (using rsa-
sha2-512)
        Key ID: "srl-user1"
        Serial: 0
        Valid: forever
        Principals:
                admin
        Critical Options: (none)
        Extensions:
                permit-X11-forwarding
                permit-agent-forwarding
                permit-port-forwarding
```

```
                permit-pty
                permit-user-rc
```

When a user logs into the server over SSH, the user certificate is checked against a list of configured principals. The server authenticates the client over SSH only if the following conditions are met: the client's certificate is verified, the client's username matches, and the client's certificate has a principal that is configured on the server. A single user (client) certificate can match multiple principals, and multiple principals can match a single user.

You can configure SSH principals under `system.aaa.authentication.user{}`.

Example

The following example configures user (`srl-user1`) with role `admin`, and ssh-principal allowing `admin`.

```
--{ candidate shared default }--[  ]--
# info with-context system aaa authentication user srl-user1
    system {
        aaa {
            authentication {
                user srl-user1 {
                    role [
                        admin
                    ]
                    ssh-principals [
                        admin
                    ]
                }
            }
        }
    }
```

3.7.3.3 Configuring host certificate

Procedure

SR Linux provides an option to configure the host certificates (`certificate`) under `system.ssh-server.host-key`. The host certificate refers to the host key signed with a CA private key. For more information about SSH host keys, see SSH host keys. In SR Linux, the server-side keys are stored in `/etc/ssh`, and the filenames begin with `ssh_host_[`*`dsa/ecdsa/ed25519/rsa`*`]_key`. The `certificate` configuration is a global setting and applicable to all SSH servers. You must configure the host to present the certificate during user login.

Example

The following example configures host certificates with `preserve` option set to `true`. This ensures the SSH server host keys in `/etc/ssh` are preserved and restored after a system reboot or SSH server restart.

```
--{ candidate shared default }--[  ]--
# info with-context system ssh-server host-key
    system {
        ssh-server {
            host-key {
                certificate ssh-rsa-cert-v01@openssh.com AAAAHHNzaC1yc2EtY2VydC12MDFAb3Blbn
NzaC5jb20AAAAgz2nzyiDaRws9fN3/p8iFd5iq40Twp4eVAQMAOSwkNAMAAAADAQABAAABgQC7UC4GHVIV/
```

```
2CY9fgxUizgN+jaFUgkT2jtaE6f/B7s885JbOiQYp19ltGhbqlrZQDcK+gtYjAnJ9wOSJKDg5dyY9HgTPWi
RbRzoy3oNp7QSVwEQds8HznOP8odIWWRrHzDl9GvbTTaH2Vy5qYkHubAmnW8XSkQS6jiP0nlcpfGVn/
x04FkZZQG8eofQ5RNwvd570YpaxAlKgR4xrDVbi7r65GVm3PCdfRBry9D/L9jAmc1Svc6WcqQG17ZOxo87IXb
BxLqEs3zcc44rN+UQ76Em0ictx1K0wgemDdJrevkNQ8YPTMuiPFYEfJWoE3DpSOdTfItgdafcJUfLtYrc
WksZNuUFmlFhgqb1wRIABPNgrwU9YbtldENJnOPxURaPgYkt6dDebsAWuIlCREf6XSQzbqVpBAo3YmkRAIRW8Ds/
8ZDd6TzmiEp8J00aD0/JlZJEZ53wxABO+rzZiLzO+BXOLJUblyW25Gi3G3AGEu+H8xLWu/MFXUAf+Lgx
TnjraMAAAAAAAAAAAAAAAIAAAARY2xhYi1zc2hfdGVzdC1zcmwAAAAVAAAAEWNsYWItc3NoX3Rlc3Qtc3JsAAAAAAAAAAD/
/////////wAAAAAAAAAAAAAAAAAAAZcAAAAHc3NoLXJzYQAAAAMBAAEAAAGBAK6OpaDPfrRpDrL7umnegq
Oh+RtAW9s+Ov2DSNXbHI4IGOJ0UfwLOv1Bpqx8VEgcMXF7J0278m3Ye1CEt6StRDygqMCvK1Hvakut4z
Nzu0hMYN1C4Y1kNU3AR594JuNesnYs+L58XC9kGgUzsE1if8fHgcQd4bIUoZnlFK3H9syIxbbFXDJ+FH+znAqpdX
+RSzFCXJYkQp2MuFHMFsOoM2vdqU4m2ASE5BwPqQbucIwp1eymOJ7ZOOFrWSGzsRyD99/y39CB/jcfgT+KY/S/Uckey
KoGC1nErfATtOIYM1jx3YtAluU2ynOEt7hA52rB+JyjOT1CQih1BrBYyBx/dReIrP9qQfRklWMhzsROxsL0+DEzQmZe
AdWRqSrfsTE8uQZdXZ0RDeUmGRwLwqpr6C4erLEXsRPDhI6cPEp91E59Sei5iBUsOnCEgddEPO8E4+27k4f2IrRp1/Zx5C7v
BxACYa4HNYBYqpmU0pv0l80KkCW91ManEYQYGZcSXjqvcQAAAZQAAAAMcnNhLXNoYTItNTEyAAABgFVR+2y3659XN7uTLL6i
+LFkcU76u8OTwZl2pKPZRurBqfQC9bLuQSmawfxmGAzg5emMzqcafGelwZeIuQjmUCSlNDyH+Yhsw6MMkadl2YIfuFFu/
zzIX3fF2reSfAPTPKkbZrijclYxo9H9gqWBGJF+A0f848gk3fCATwq6JM4F3q4J0lRibQBulOuvXVm3GZVWqHou9rpHOL0aGWhu
DjTaq5/oZ+tv4X+hfrCKmkacesNeX4wNn6k3JBFVLnNxdPDwAlOcSmvPqmZHa8R6e+qvu5B44YSVmlAHsaUyYHDbVjH6Hhz
KoXGTlv3iMoNl/bWNkiCS+u4Licleh0pqhinsIi8FtdVpEXQW+unqd93XQ/hAuvxZ9ia0gugkGMQixGlbPfFTaQqkZBbv0r
+S44NpsSVPOiYw6bJ/iHZM4zFHK6l2+eGgJ4p/kx/3TSO10fHK/NaoFiZLws4geYpl3YZfxmk/KKWBctv7tlv/ySzavQQJRg
+SvbH9FibEFiRHqQ== user@node
                preserve true
            }
        }
    }
```

3.7.3.4 Trusting CA public key

To prevent the warning about an unknown host, the user (client) system must trust the CA public key that signed the host certificates. This is done by adding the CA public key to the `known_hosts` file. For example,

```
echo "@cert-authority * $(cat ca.pub)" >> ~/.ssh/known_hosts
```

where,

- **ca.pub**: references the public key of the signing CA.
- *****: specify the server being connected to or a glob value (`"*.example.com","*.mydomain.com"`

CA keys can be marked as trusted globally in the `/etc/ssh/known_hosts` file or per-user in the `~/.ssh/known_hosts` file.

3.7.3.5 Configuring revoked key list

Procedure

SR Linux provides an option to configure revoke key (`revoked-keys`) list under `.system.ssh-server`. A list of user public keys that must be invalidated is called a revoke list. Each line in the revoke list contains one public key. If a user's public key is listed in the revoke list, the server rejects the connection when the user tries to connect using SSH.

Example

The following example configures the revoked key list.

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

```
# info with-context system ssh-server revoked-keys
    system {
        ssh-server {
            revoked-keys [
                "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDPlZ4/NVdfAgaUcA2Mepy/tOYvWWhiYOmPsc
WzvG8+aj9Ssj88pmJGb0r/e3FBYxEK8Bj1Mjar7GNbUXOt3NvCLjcPo6i/xxdouA+N8RmxpbPFLsLC6JwZq5IkNoxdJ9PLUNz
+cgLOY2XiLjgrUOsCT02SOn/mjcIAxzFkqmf9eLNW9CLqBW7hVCVwHpzj4Ep814MmJBXZX9llGVj1sule+ajfUxotlChyScF3Nbd+h
+YMvfAgngFRuCebMG6n2vr+zRlB/TRQibAO5fKoi/GEuntnv4f/iaNLhlRQitIDzXFNW7xx5yIer30FnYf6u5y1ijA5XJwHb9Eos
RjoM7PBYhdC6XESuyS7gOji3/l/wQfLZ7twMLB5MxUL4aB5SnaRlkYX14GyIdpsvtoe8mfsy011PqMWmQjZ8lm2eUHN+68UNBod
MmzGLS9I11a1YODydV9gLD3WwgHxGQ1vOrGvp401S4HF26gAXrAr2QblQ7tJtq+34Fvz8O0YnplvCoU= user@node"
            ]
        }
    }
```

3.7.3.6 Viewing an SSH certificate

Procedure

You can view an SSH certificate using the following command:

```
# ssh-keygen -L -f
```

where,

- **L**: lists the content of the certificate
- **f**: specifies the path to the signed certificate

The SSH certificate formats and semantics are as per the specification defined in PROTOCOL.certkeys.

Example: Viewing an user SSH certificate

The following command displays the user SSH certificate. User SSH certificate is used to authenticate users to servers.

```
# ssh-keygen -L -f user1-cert.pub
user1-cert.pub:
        Type: ssh-rsa-cert-v01@openssh.com user certificate
        Public key: RSA-CERT SHA256:LBPq+l+ksZi++61XoVFGch5Sf3xrksQ5mY70TR3iai4
        Signing CA: RSA SHA256:GjGiO2dJV2tKBcuCjC5skMn8cfJ/iToZ/4oR/F0hhAE (using rsa-
sha2-512)
        Key ID: "user1"
        Serial: 0
        Valid: forever
        Principals:
                admin
        Critical Options: (none)
        Extensions:
                permit-X11-forwarding
                permit-agent-forwarding
                permit-port-forwarding
                permit-pty
                permit-user-rc
```

Example: Viewing a host SSH certificate

The following command displays the host SSH certificate. Host SSH certificate is used to authenticate server hosts to users.

```
# ssh-keygen -L -f ssh_host_rsa_key-cert.pub
ssh_host_rsa_key-cert.pub:
        Type: ssh-rsa-cert-v01@openssh.com host certificate
        Public key: RSA-CERT SHA256:NgxyhWgpt4BX+eglv10JVZyweD7DAdjGs0ygBHzR7Ko
        Signing CA: RSA SHA256:GjGiO2dJV2tKBcuCjC5skMn8cfJ/iToZ/4oR/F0hhAE (using rsa-
sha2-512)
        Key ID: "srl"
        Serial: 0
        Valid: forever
        Principals:
                srl
                srl.local
        Critical Options: (none)
        Extensions: (none)
```

3.7.3.7 Configuring SSH port forwarding

Procedure

You can enable port forwarding for one or more network instances on the SR Linux device. Enabling SSH port forwarding tunnels traffic securely over an encrypted SSH connection. One of the possible use case of port forwarding is to allow traffic to be sent from a local port on the SSH client to a port on the SSH server instance on the remote SR Linux device.

Example

In the following configuration example, SSH port forwarding is enabled for the SSH server instance (*srl_ssh_server*). The SR Linux device listens for SSH connections on port 8022, and the `port-forwarding true` statement allows SSH clients that log in to this server to create secure port-forwarding tunnels.

```
--{ + candidate shared default }--[  ]--
# info with-context system ssh-server srl_ssh_server
    system {
        ssh-server srl_ssh_server {
            admin-state enable
            network-instance mgmt
            port 8022
            port-forwarding true
            source-address [
                199.0.0.1
            ]
        }
    }
```

3.7.4 Configuring FTP

Procedure

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 with-context system ftp-server
    system {
        ftp-server {
            network-instance mgmt {
                admin-state enable
                source-address [
                    192.0.2.1
                ]
            }
            network-instance default {
                admin-state enable
                source-address [
                   192.0.2.4
                ]
            }
        }
    }
```

3.8 Configuring banners

Procedure

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 into the system, and the motd banner text appears after the user has been authenticated.

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

```
    system {
        banner {
            login-banner "Enter your SR Linux login credentials."
            motd-banner "Welcome to the SR Linux CLI. Your activity may be monitored."
        }
    }
```

3.9 Synchronizing the system clock

Procedure

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. The NTP server used in the system NTP client configuration can be IP address-based or hostname-based.

Example: Enabling an NTP client with an IP address-based NTP server

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

The example uses an IP address for the NTP server.

```
--{ candidate shared default }--[  ]--
# info with-context system ntp
    system {
        ntp {
            admin-state enable
            network-instance mgmt
            server 4.53.160.75 {
            }
        }
    }
```

Example: Enabling an NTP client with a hostname-based NTP server

The following example enables the NTP client on the SR Linux device and sets a hostname-based NTP server for clock synchronization.

```
--{ candidate shared default }--[  ]--
# info with-context system ntp
    system {
        ntp {
            admin-state enable
            network-instance mgmt
            server srlinux.nokia.com {
            }
        }
    }
```

For the hostname support to work,

- DNS configuration is mandatory for this to function correctly. A missing configuration does not prevent configuration; however, the NTP session to that server can only become active when DNS is functioning correctly.
- The network-instance of the NTP server and DNS must be the same.

3.10 Configuring preferred source address for NTP requests

Procedure

SR Linux allows you to specify the preferred source address when acting as an NTP client in a `network-instance` with multiple possible source addresses. The specified source IP address must belong to the network instance where NTP is configured. You can configure the preferred `source-address` parameter as either an IPv4 or IPv6 address under `system.ntp`.

In this example, NTP requests are sent to the configured NTP servers using the source address of 4.53.160.75.

Example

In this example, NTP requests are sent to the configured NTP servers using the source address of 4.53.160.75.

```
--{ candidate shared default }--[  ]--
# info with-context system ntp
    system {
        ntp {
            network-instance mgmt
            source-address 4.53.160.75
            server 129.6.15.28 {
            }
            server 216.239.35.0 {
            }
        }
    }
```

3.11 NTP TLS support

SR Linux supports the following authentication methods for trusted communication with NTP servers over both IPv4 and IPv6:

- Network Time Security (NTS)
- Message Authentication Code (MAC) symmetric keys (NTP key authentication)

Depending on the operator configuration, `nts` and `ntp-key` can be defined under the same server entry. The system then creates two separate server entries, one for each type of authentication.

3.11.1 Configuring NTP TLS (NTS) authentication

Procedure

You can configure NTS on a per-server basis. The NTS Key Establishment (NTS-KE) protocol uses Transport Layer Security (TLS) to obtain the necessary keys and cookies for the NTP packet authentication.

For each NTS certificate, a TLS profile (**tls-profile**) can be linked and configured as a standard TLS profile within the `system.ntp.server` context.

If a TLS profile is not associated, the default system CA certificates are used.

When a TLS profile is configured, SR Linux checks if a different server entry has used this TLS profile. If not, a new certificate set value is added that points to the certificate location on the file system. If the TLS profile has been previously associated, a reference can be set for each configured server entry.

Example: Configuring NTP TLS (NTS) authentication

The following example configures the NTS authentication method to secure the time synchronization with an NTP server (`srlinux.nokia.com`).

```
--{ * candidate shared default }--[  ]--
# info with-context system ntp server srlinux.nokia.com
    system {
        ntp {
            server srlinux.nokia.com {
                nts true
                tls-profile clab-profile
            }
        }
    }
```

Note:
NTS key exchange requires an ACL rule to allow TCP traffic with a source port value of 4460.

3.11.2 Configuring NTP key authentication

Procedure

The NTP protocol supports MAC to prevent system time tampering.

An SR Linux operator can configure NTP key authentication using the **system.ntp.ntp-key** command, which requires defining the **key-ID**, **key-type**, and **key-value** parameters.

The **key-ID** option specifies the key (with a value between 1 and 2^32-1) that chronyd uses to authenticate requests sent to the server and verify its responses.

Note: SR Linux supports **key-ID** values **only** within the range of 1 to 65,535.

SR Linux supports the following key types (**key-type**):

- NTP_AUTH_MD5
- NTP_AUTH_SHA1

- NTP_AUTH_SHA256
- NTP_AUTH_SHA384
- NTP_AUTH_SHA512
- NTP_AUTH_AES_CBC_128
- NTP_AUTH_AES_CBC_256

Example: Configuring NTP key authentication

The following example configures NTP and NTP key authentication methods to secure the time synchronization with an NTP server (`srlinux.nokia.com`).

```
--{ * candidate shared default }--[  ]--
# info with-context system ntp
    system {
        ntp {
            network-instance default
            server srlinux.nokia.com {
                key-id 700
                nts true
                tls-profile clab-profile
            }
            ntp-key 700 {
                key-type NTP_AUTH_SHA256
                key-value 2e834852e7e41eda65b258
            }
        }
    }
```

3.12 Configuring SNMP

Procedure

To configure SNMP, enable an SNMP server on one or more network-instances. The SR Linux device supports SNMPv2. The MIB file that covers these OIDs is packaged with each release.

See the *SR Linux System Management Guide* for the procedure for configuring an SNMP server.

3.13 Powering down the system

About this task

The SR Linux device does not feature a power switch. If you need to power down the device, use the following procedure.

Procedure

Step 1. Enter the CLI command to reboot the system.

Example

```
# tools platform chassis reboot
```

Step 2. When the GRUB menu appears, remove the power cables from the PSUs.

3.14 Disabling the USB interface

Procedure

The USB interface is enabled by default at run-time and can be disabled in configuration using the **platform control interface usb admin-state** command.

Notice: This command only affects the run-time USB state and does not apply to boot time or the recovery process.

Example: Disable USB interface

The following configuration example disables the USB interface on control module slot A , which prevents the USB device from being initialized and the system from loading any drivers.

```
--{ * candidate shared default }--[  ]--
# info with-context platform control A
    platform {
        control A {
            interface usb {
                admin-state disable
            }
        }
    }
```

3.15 Configuring reboot options

You can perform a reboot using the **tools platform** <*component*> {*slot*} command. The **reboot** command is supported for all platform components, namely chassis, control (active/standby), fabric, and linecard slots.

This command triggers an immediate reboot.

- chassis: **tools platform chassis reboot**
- control slot: **tools platform control <slot> reboot**
- fabric slot: **tools platform fabric <slot> reboot**
- linecard slot: **tools platform linecard <slot> reboot**

3.15.1 delay

You can use the **delay** option to set a wait time before rebooting. The delay period required is configured in seconds. During this period, you can cancel any pending reboot operations.

Note: When a delayed reboot is pending for either a chassis or an active control component, setting a wait time to delay the reboot of any other component (linecard, standby control, or fabric slot) is not supported.

- chassis: **tools platform chassis reboot delay <value>**

```
# tools platform chassis reboot delay 2
/platform:
 chassis will reboot in 2 seconds
```

- control slot: **tools platform control <slot> reboot delay <value>**
- fabric slot: **tools platform fabric <slot> reboot delay <value>**
- linecard slot: **tools platform linecard <slot> reboot delay <value>**

When doing a warm redundancy CPM switchover, any pending delayed reboot of active or standby control card is discarded. The command `.tools.platform.redundancy.switchover` is not impacted by any pending delayed reboots.

3.15.2 cancel

You can use the **cancel** option to cancel any pending reboot on a platform component. When there are no pending delayed reboots, the reboot cancel command execution fails.

- chassis: **tools platform chassis reboot delay cancel**

```
# tools platform chassis reboot cancel
The chassis reboot has been canceled
/platform:
    chassis reboot has been canceled
```

- control slot: **tools platform control <slot> reboot cancel**
- fabric slot: **tools platform fabric <slot> reboot cancel**
- linecard slot: **tools platform linecard <slot> reboot cancel**

Note: You cannot combine the **cancel** option with **force** or **delay** options.

3.15.3 force

You can use the **force** option to trigger a force reboot of a platform component. This option is supported only for the chassis and control slots. The **force** option overrides all synchronization activities, and any soft checks that prevent a reboot (for example, the unsaved configuration between running and startup) are ignored.

Caution: Forcing a reboot immediately after an image change may result in a standby module booting an older image.

- chassis: **tools platform chassis reboot force**
- control slot: **tools platform control <slot> reboot force**

3.15.4 message

You can use the **message** option to broadcast a user-defined message to other users before the reboot occurs. The message option can be used with reboot types of immediate, cancel, or delay.

- chassis: **tools platform chassis reboot message <value>**

```
#  tools platform chassis reboot message "This is a message"
 chassis is rebooting now: "This is a message"
```

- control slot: **tools platform control <slot> reboot message <value>**
- fabric slot: **tools platform fabric <slot> reboot message <value>**
- linecard slot: **tools platform linecard <slot> reboot message <value>**

The following examples demonstrate different reboot scenarios with the **message** option:

- Immediate reboot with the **message** option.

```
# tools platform chassis reboot message "this is a message"
 chassis is rebooting now: "this is a message"
```

  A message indicating the initiation of an immediate reboot is broadcast to all users only if the immediate reboot command is executed with the **message** option.

- Delayed reboot with the **message** option.

```
# tools platform control A reboot delay 100 message "this is a message"
 control slot A is rebooting in 100 seconds (at 2023-01-24T01:45:20.556Z): "this is a
 message"
/platform/control[slot=A]:
    control slot A will reboot in 100 seconds
```

  A message indicating the reboot delay is broadcast to all users only if the reboot delay command is executed with the **message** option.

- When a pending delayed reboot is executed for an active control card or the chassis, the user-defined message is broadcast, and the prompt is also updated accordingly.

```
# tools platform control A reboot delay 3600 message "control reboot message"
/platform/control[slot=A]:
 control slot A will reboot in 3600 seconds
 --{ [ACTIVE CONTROL REBOOT IN 3599 SECONDS (control reboot message)] running }--[ ]--
```

```
# tools platform chassis reboot delay 3600 message "reboot message"
/platform:
 chassis will reboot in 3600 seconds
 --{ [SYSTEM REBOOT IN 3599 SECONDS (reboot message)] running }--[ ]--
```

- Canceling a delayed reboot with the **message** option.

```
#tools platform linecard 1 reboot cancel message "this is a cancel message"
 linecard slot 1 reboot has been canceled: "this is a cancel message"
```

```
/platform/linecard[slot=1]:
 linecard slot 1 reboot has been canceled
```

The preceding command cancels all the pending delayed reboots and broadcasts the user-defined message. A message indicating the reboot cancel is broadcast to all users only if the reboot cancel command is executed with the **message** option.

- When a delayed reboot time expires and the reboot is about to happen, the following message is broadcast:

```
chassis: chassis is rebooting now
control: control slot <A or B> is rebooting now
linecard: linecard slot <x> is rebooting now
fabric: fabric slot <x> is rebooting now
```

Note: This message is broadcast irrespective of executing the reboot delay command with or without the **message** option.

- When a delayed reboot fails, the following message is broadcast:

```
chassis: chassis reboot failed: "<reason for failure>"
control: control slot <A or B> reboot failed: "<reason for failure>"
linecard: linecard slot <x> reboot failed: "<reason for failure>"
fabric: fabric slot <x> reboot failed: "<reason for failure>"
```

Note: This message is broadcast irrespective of executing the reboot delay command with or without the **message** option.

- When a delayed reboot is pending for a component, and a reboot (delay or immediate) is attempted for a chassis or active control, the reboot gets rejected with the following message:

```
chassis: disallowed, delayed reboot is pending for linecard slot 1
active control slot A: disallowed, delayed reboot is pending for linecard slot 1
```

This example shows a reboot attempted on a linecard when control A is active.

Note: The error message lists only the first found component with a delayed reboot pending. In this example, if all linecards <1 to 4> are pending delayed reboots, the error message highlights only the pending reboot for linecard1.

- When a delayed reboot is pending for a chassis or active control, and a reboot (delay or immediate) is attempted for any other platform component, the reboot gets rejected with the following error message.

  This example shows a reboot attempted on a fabric slot when a reboot is pending for the active control module A.

```
fabric slot <x>: disallowed, delayed reboot is pending for control slot A
```

This example shows a reboot attempted on a fabric slot when reboot is pending for chassis component.

```
fabric slot <x>: disallowed, delayed reboot is pending for chassis
```

- While a delayed reboot is pending for a component, and a reboot (delay or immediate) is attempted for the same platform component, the reboot gets rejected with a message indicating that the pending delayed reboot must first be cancelled.
- When the wait time specified for the reboot delay exceeds the maximum delay limit, the delayed reboot command gets executed and the following message is displayed:

```
# tools platform linecard 1 reboot delay 18446744073709551615
/platform/linecard[slot=1]:
 delay has been limited to 16772217903 seconds
 /platform/linecard[slot=1]:
 linecard slot 1 will reboot in 16772217903 seconds
```

3.15.5 warm

When you execute the reboot command with the **warm** option, it validates the current configuration and prompts reboot confirmation. On confirming, the system reboots without impacting the datapath. If a warm reboot is performed after a new image is configured, the system upgrades to the new image.

Before performing a warm reboot, you must confirm if the current SR Linux configuration and state supports warm reboot. Use the **tools platform chassis reboot warm validate** command.

```
--{ running }--[  ]--
A:# tools platform chassis reboot warm validate
/platform:
    Warm reboot validate requested
 /:
    Success
```

If the validation is successful, proceed with the warm reboot.

If the validation is unsuccessful, or if an attempt to perform a warm reboot fails, you can force the warm reboot using the additional **force** option.

Caution: Forcing a warm reboot may result in a service outage. The **force** option overrides any warnings, such as peers that are not configured or peers that do not support graceful restart.

```
--{ running }--[  ]--
A:# tools platform chassis reboot warm force
/platform:
    Warm reboot force requested

/:
    Success
```

See the “Configuration state support” section in the *SR Linux Software Installation Guide* for information about how to use warm reboot during an ISSU.

3.16 Non-Stop Forwarding

Note: This feature is currently supported on 7250 IXR (including mixed), 7250 IXR Gen 2, 7250 IXR Gen 2c+, and 7250 IXR Gen 3 platforms.

Non-Stop Forwarding or NSF, is the sequence of processes required to effectively switch control of a running system between two CPMs (active/standby), without disrupting the data forwarding. It allows the router to continue forwarding data with previously known route/state information, while the control plane restarts and re-converges.

Similar to system warm reboot, NSF depends on graceful restart helpers, but it is primarily used for unplanned outage (for example, control plane failover) and cannot be used for upgrades.

During an NSF switchover, no control plane or management plane functions are available, including refreshing of neighbors, and slow path functions like DHCP relay and responding to ARP/ND.

In SR Linux, NSF leverages application warm restart, with the fundamental design to synchronize the IDB server (`idb_server`) with the standby CPM, and to allow applications to leave their state information in IDB during the restart, then recover it after the restart. For more information, see Triggering redundancy switchover and Forcing redundancy synchronization.

NSF features and limitations

NSF is supported with the following feature set:

- Supports ACL, IPv4, and IPv6
- Supports complete QoS feature set including queue configuration, classifiers, ECN with counters cleared.
- Supports IPv4/IPv6 routing.
- Supports BGP with IPv4/IPv6-unicast address families, where all neighboring devices support graceful restart helper.

  Note: Peers who do not support graceful restart withdraw routes during outages, impacting the data path for traffic destined for the system undergoing NSF. However, NSF must be attempted for any peers that support graceful restart helper.

- Supports IS-IS with IPv4/IPv6-unicast address families, where all neighboring devices support graceful restart helper.

  Note: Peers who do not support graceful restart withdraw routes during outages, impacting the data path for traffic destined for the system undergoing NSF. However, NSF must be attempted for any peers that support graceful restart helper.

- Supports LAG and LACP with both slow and fast timers.
- Supports P4RT
- Supports gRIBI. During NSF, though management plane functions are unavailable, gRIBI programmed routes are persisted.
- Supports sFlow on slow path.
- Supports LLDP.

Note: In case of session time out, adjacency loss may occur.

3.16.1 Triggering redundancy switchover

Procedure

Use the command `tools platform redundancy switchover` to trigger a redundancy switchover from active to the standby control module. The switchover happens in conjunction with a cold restart.

Example

To trigger redundancy switchover:

```
--{ running }--[  ]--
# tools platform redundancy switchover
```

The NSF implementation changes the default behavior of this command execution. Upon NSF implementation, when a redundancy switch over is triggered, the system always attempts to perform a NSF failover in conjunction with a warm restart.

Note: The command `tools platform redundancy switchover` is not impacted by any pending delayed reboots.

Note:
The **tools.platform.chassis.redundancy.switchover** command requires all AHR-capable apps to be in a synced state before triggering a switchover. The synced state indicates that both file system and IDB synchronization have completed (as per NSF), and all supported protocols are ready for a HA swithchover. If the system is not ready for a switchover (not all AHR-capable apps are synced), the switchover is not be triggered, and the user is notified with a reason (**.platform.chassis.redundancy.synchronization.state-reason**)

3.16.2 Forcing redundancy synchronization

Procedure

You can use the **tools platform redundancy synchronize overlay system** to synchronize the overlay file system or system-required data between the active and standby control modules. The NSF implementation extends this behavior by including the synchronization of file system and IDB server info between the modules.

Example

To synchronize, use the following command:

```
--{ running }--[  ]--
# tools platform redundancy synchronize overlay system
```

3.17 Non-Stop Routing

SR Linux routers support a Non-Stop Routing (NSR) for control plane High Availability (HA).

This is implemented through a mechanism called Application Hot Restart (AHR), which allows individual applications to restart rapidly in the event of a failure, minimizing downtime and ensuring continuous operation.

Note: AHR is currently supported only for the **`isis_mgr`** application and is available on the 7250 IXR (including mixed), 7250 IXR Gen 2, 7250 IXR Gen 2c+, and 7250 IXR Gen 3 platforms.

AHR enables individual control plane applications to recover from failures without disrupting active sessions or affecting the overall control plane stability. Even if an application fails on the active control module, the control plane continues to operate without interruption. On systems with redundant control module , a failure in a specific control plane application does not impact overall route functionality or control plane operation.

Applications that do not support AHR rely on Application Warm Restart (AWR) to maintain HA. For information about supported restart types, see Restarting applications.

3.17.1 NSR on redundant control systems

On chassis-based systems, AHR is always enabled and cannot be disabled. If an application crashes and the standby unit fails to synchronize, meaning the redundancy state of the platform **platform.redundancy.control-plane.synchronization.state** is not set to **synchronized**, the `app_mgr` handles recovery using its standard failure process. The application is warm-restarted, if the application supports it; otherwise, the application is cold-restarted.

To trigger a switchover, the **tools.platform.chassis.redundancy.switchover** command requires all AHR-capable applications to be in a synchronized state. The synchronized state indicates that both file system and IDB synchronization are completed (as per NSF), and all supported protocols are ready for a HA switchover.

If the system is not ready for a switchover (not all AHR-capable apps are synchronized), the switchover is not triggered, and the system provides a reason (**.platform.redundancy.control-plane.synchronization.state-reason**).

In an emergency, the only way to recover is to reboot the active CPM. Rebooting the active CPM causes all applications to restart cold.

For AHR-capable applications, the name includes an `_A` or `_B` suffix .

For example, `isis_mgr_A` and `isis_mgr_B`.

On redundant systems, `isis_mgr_A` runs on CPM A and `isis_mgr_B` on CPM B. Either of these can be the active instance, depending on which CPM is active. The `isis_mgr` instance running on the active CPM is the active application instance.

When an application is hot-restarted, the instance on the active CPM is hot-started and the instance on the standby CPM is cold-restarted without impacting forwarding.

3.17.2 Displaying AHR application synchronization state

Procedure

You can display the synchronization state of an application using the **info.from.state.platform.redundancy control-plane.synchronization** command.

Example: Display application synchronization state

The following configuration example displays the application synchronization state.

```
--{ candidate shared default }--[  ]--
# info with-context from state platform redundancy control-plane synchronization
    platform {
        redundancy {
            control-plane {
                synchronization {
                    state synchronized
                    last-synchronization "2025-06-12T13:11:04.000Z (8 minutes ago)"
                    overlay {
                        synchronization-frequency 60
                    }
                }
            }
        }
    }
```

Example: Display application synchronization state and failure details

The following configuration example displays the application synchronization state and failure details.

```
--{ candidate shared default }--[  ]--
# info with-context from state platform redundancy control-plane
    platform {
        redundancy {
            control-plane {
                synchronization {
                    state not-ready
                    state-reason "not synchronized: filesystem; database; applications"
                    last-synchronization "2025-06-12T13:11:04.000Z (14 minutes ago)"
                    overlay {
                        synchronization-frequency 60
                    }
                }
            }
        }
    }
```

3.18 VRRP

Virtual Router Redundancy Protocol (VRRP) describes a method of implementing a redundant IP interface shared between two or more routers on a common LAN segment, allowing a group of routers to function as one virtual router. When this IP interface is specified as a default gateway on hosts directly attached to a shared broadcast domain, the routers sharing the IP interface prevent a single point of failure by limiting access to this gateway address.

VRRP for IPv4 is defined in the IETF RFC 3768, *Virtual Router Redundancy Protocol*. VRRP version 3 for both IPv4 and IPv6 is defined in RFC 5798.

Supported platforms

VRRP is supported on the following platforms:

- 7250 IXR Gen 2c+
- 7250 IXR Gen 3
- 7730 SXR

VRRP election process

VRRP routers use VRRP advertisement messages to communicate. These messages include a VRRP priority value. The router with the highest configurable priority value becomes the active VRRP router and assumes control of the virtual IP address for the VRRP instance. If multiple routers have the same value, the router with the highest IP address becomes the active VRRP router.

If the active router fails to send advertisements within a certain time frame, a new election occurs, and the highest priority backup router becomes the active router.

3.18.1 Configuring VRRP

About this task

To configure VRRP within an IP subinterface context, you must configure a virtual router ID using the **vrrp-group** command. To specify the IP address for the associated virtual router, configure the **virtual-address** parameter.

The following list outlines key optional parameters:

- **priority** – configures VRRP router priority to a specific value
- **advertise-interval** – configures VRRP message frequency
- **preempt** – controls whether a new VRRP router with a preferred VRRP priority can assume the role of the active VRRP
- **authentication** – configures authentication for VRRP messages

The following example shows a sample VRRP (V4 and V6) instance configuration.

Example

```
--{ + candidate shared default }--[ interface ethernet-1/4 ]--
#  info
    admin-state enable
    vlan-tagging true
    ethernet {
        port-speed 40G
    }
    subinterface 1 {
        admin-state enable
        ipv4 {
            admin-state enable
            address 10.1.1.1/24 {
                vrrp {
                    vrrp-group 1 {
                        admin-state enable
```

```
                        priority 110
                        preempt true
                        accept-mode true
                        virtual-address [
                            10.1.1.100
                        ]
                    }
                }
            }
            address 10.1.2.1/24 {
                vrrp {
                    vrrp-group 2 {
                        admin-state enable
                        priority 110
                        preempt true
                        accept-mode true
                        virtual-address [
                            10.1.2.100
                        ]
                    }
                }
            }
            address 10.1.3.1/24 {
                vrrp {
                    vrrp-group 3 {
                        admin-state enable
                        priority 110
                        preempt true
                        accept-mode true
                        virtual-address [
                            10.1.3.100
                        ]
                    }
                }
            }
        }
        ipv6 {
            address 2001:db8:10:1:1::1/64 {
                vrrp {
                    vrrp-group 1 {
                        admin-state enable
                        priority 110
                        preempt true
                        accept-mode true
                        virtual-address [
                            2001:db8:10:1:1::100
                        ]
                    }
                }
            }
        }
        vlan {
            encap {
                single-tagged {
                    vlan-id 1
                }
            }
        }
    }
```

3.19 Switching modes for Ethernet frame forwarding

7220 IXR-D*x* platforms support two switching modes for Ethernet frame forwarding:

- store-and-forward (SF)
- cut-through (CT)

Store-and-forward mode

In SF mode, the system waits to receive the entire Ethernet frame, including the 4-byte Cyclic Redundancy Check (CRC) at the end, before making any forwarding decisions. The frame is only forwarded if the CRC is valid. If the CRC check fails, the frame is discarded, and an error counter is incremented.

Cut-through mode

In CT mode, the system begins forwarding the frame as soon as it reads the Layer 2 or Layer 3 headers, without waiting for the entire frame to be received. If the target egress queue of the egress port is not congested, the bytes of the received frame are transmitted across the switch fabric as they are received. The fully intact frame is transmitted from the egress port as soon as all the bytes are received.

7220 IXR-D*x* platforms support CT mode, with SF as the default setting for backward compatibility. The system allows switching between SF and CT modes using the **system datapath forwarding-mode** command.

Even when CT mode is enabled, the following conditions force the system to revert to SF mode for specific frames:

- packets bound for or originating from the Control Plane Module (CPM)
- packets received or sent on management ports
- packets crossing speed mismatches (especially with a greater than 4:1 ratio)
- multicast frames
- frames targeting congested queues

The **info from state interface forwarding-mode** command displays the active forwarding mode on each port. In CT mode, all non-management ports display the forwarding mode as cut-through. The statistics that indicate the number of packets or frames received or transmitted on a port, and that were forwarded using the CT mode, are not available.

Interaction with flow control and Priority Flow Control (PFC)

When the system operates in CT mode:

- the processing of received non-PFC PAUSE frames is automatically disabled on all ports
- PFC frames work normally in both receive and transmit directions

3.19.1 Configuring switching mode for Ethernet frame forwarding

Procedure

To configure a switching mode for Ethernet frame forwarding on 7220 IXR-D*x* platforms, specify the mode using the **system datapath forwarding-mode** command. Use the **info from state interface forwarding-mode** command to display the active switching mode on an interface.

Example: Switching mode configuration

The following example shows the switching mode configuration.

```
--{ + candidate shared default }--[  ]--
# info with-context system datapath
    system {
        datapath {
            forwarding-mode cut-through
            icmp {
                rate-limit-per-host {
                    peak-rate 3
                    max-burst 3
                }
            }
        }
    }
```

Example: Display active switching mode on an interface

The following example displays the active switching mode on an interface.

```
--{ + candidate shared default }--[  ]--
# info with-context from state interface ethernet-1/1 forwarding-mode
    interface ethernet-1/1 {
        forwarding-mode store-and-forward
    }
```

4 Configuration management

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

4.1 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 the device into management, enables DHCP/v6 on the management interface, adds it to the management network-instance, enables an SSH server, 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.

4.2 Configuration datastores

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

- **running**

  The running datastore contains the currently active configuration.

- **state**

  The state datastore contains the running configuration, plus dynamically added data such as the operational state of interfaces or BGP peers added via auto-discovery, as well as session states and routing tables.

- **candidate**

  The candidate datastore contains a user-configurable version of the running datastore. After it has been committed, the candidate datastore becomes the running datastore.

- **tools**

  The tools datastore contains executable commands that allow you to perform operations such as restarting the device and clearing interface statistics.

In 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 the configuration from the running datastore for the current context.

4.3 Configuration modes

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

4.3.1 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**

  The shared configuration candidate is the default 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**

  The private configuration candidate is the default 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.

4.3.2 Setting the configuration mode

Procedure

After logging in to the CLI, you are initially placed in running mode. From running mode, you can change to a configuration mode. Table 4: Commands to change configuration mode describes the commands to change between modes.

Table 4: 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>**

To enter this mode:	Type this command:
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: 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: Enter candidate mode for a named configuration candidate

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

4.4 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. Only use the **commit** command in candidate mode.

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.

4.4.1 Confirming a commit operation

Procedure

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: Commit the configuration and start the confirmation timer

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: Confirm the commit operation and make the configuration changes permanent

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: Revert to the previous configuration

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
```

4.4.2 Validating a commit operation

Procedure

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: 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.

4.4.3 Updating the baseline datastore

Procedure

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 is updated automatically when the user commits the changes to the configuration or can be updated 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 exclamation mark (!) 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 }--[  ]--
#
```

4.5 Deleting a configuration

Procedure

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 with-context system banner
    system {
        banner {
            login-banner "Welcome to SR Linux!"
        }
    }
--{ candidate shared default }--[  ]--
# delete system banner
--{ candidate shared default }--[  ]--
# info with-context system banner
    system {
        banner {
        }
    }
```

4.6 Annotating the configuration

Procedure

To aid in reading a configuration, you can add comments or descriptive annotations. The annotations are indicated by three exclamation marks (! ! !) 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: Add a comment to a configuration

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 acl-filter ip_tcp type ipv4 !! "Filter TCP traffic"
```

Example: Replace an existing comment

To replace the existing comment, use three exclamation marks (**! ! !**) instead of two exclamation marks (**! !**) 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 acl-filter ip_tcp type ipv4
--{ * candidate shared default }--[ acl acl-filter ip_tcp type ipv4 ]--
# 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 press Enter.

Example: Display a comment in context

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

```
--{ running }--[  ]--
# info with-context acl
    acl {
        acl-filter ip_tcp type ipv4 {
            !!! Filter TCP traffic
            entry 1000 {
                description "Match IP Address TCP Protocol Ports"
                match {
                    ipv4 {
                        protocol tcp
                        destination-ip {
                            prefix 10.1.3.1/32
                        }
                        source-ip {
                            prefix 10.1.5.1/32
                        }
                    }
                    transport {
                        destination-port {
                            value 6789
                        }
                        source-port {
                            value 6722
                        }
                    }
                }
                action {
                    log true
                    accept {
                    }
                }
            }
        }
```

```
    }
```

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

4.7 Discarding a configuration in candidate mode

Procedure

You can discard previously applied configurations with the **discard** command. Only use the **discard** command in candidate mode.

- 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 ]--
```

4.8 Displaying configuration details

Procedure

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: Display the entire configuration

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

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

Example: Display the configuration for a specific context

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: Display the configuration within a context

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: Display the configuration as JSON-formatted output

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: Display values of all parameters in the configuration

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: Display the configuration as set statements

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: Display a specified number of sub-context levels

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
    }
```

4.9 Displaying the configuration state

Procedure

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

Example: Display state information for a specified context from candidate or running mode

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

Example: Display state information for a specified context from state mode

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

Example: Change to a different mode while remaining in the previous context

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 ]--
```

4.10 Saving a configuration to a file

Procedure

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

Example: 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: Save the running configuration to a JSON-formatted file

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

Example: 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.

4.11 Loading a configuration

Procedure

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: 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: Load a configuration from a file

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

```
Loading configuration from 'home/config.txt'
```

Example: 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: 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 SR Linux!"
<< }
```

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

4.12 Executing configuration statements from a file

Procedure

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). 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.

4.13 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.

4.13.1 Generating a checkpoint

Procedure

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: Generate 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 ''
```

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: Generate a checkpoint automatically

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

```
--{ !* candidate shared default }--[  ]--
# info with-context 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.

4.13.2 Loading a checkpoint

Procedure

To load a checkpoint, enter candidate mode and specify the checkpoint to load.

Example

The following example loads a checkpoint into the candidate configuration.

Note: 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'
```

4.13.3 Reverting to a previous checkpoint

Procedure

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
```

4.13.4 Clearing a checkpoint

Procedure

You can clear checkpoints from the system manually with a **tools** command.

In addition, checkpoints can be cleared 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.

```
--{ * candidate shared default }--[  ]--
# tools system configuration checkpoint 2 clear
/system/configuration/checkpoint[id=2]:
    Cleared checkpoint '/etc/opt/srlinux/checkpoint/checkpoint-2.json'
```

4.13.5 Configuring maximum number of checkpoints

Procedure

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 with-context 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.

4.13.6 Displaying checkpoint information

Procedure

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

Example

```
# info with-context 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
            }
        }
    }
```

4.14 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` file fails when the system starts.

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

4.14.1 Saving a rescue configuration

Procedure

You can save a rescue configuration to be loaded automatically if the default `config.json` file 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: Save a rescue configuration

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'
```

The following **save rescue** command also saves a rescue configuration:

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

Example: List contents of the checkpoint directory

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
```

4.14.2 Clearing a rescue configuration

Procedure

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'
```

You can confirm the rescue configuration is cleared by viewing the checkpoint directory; for example:

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

4.15 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

4.15.1 Upgrading configuration files

Procedure

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

Example: Upgrade the startup configuration file

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

Example: Upgrade the rescue configuration file

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

Example: Upgrade a configuration checkpoint file

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

Example: Upgrade a specified JSON-formatted configuration file

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

5 Securing access

The SR Linux device 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 device 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+ or RADIUS).

5.1 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.

5.1.1 Linux users

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

By default, the SR Linux has a single Linux user, `linuxadmin`, who has access to `sudo` to root and can run the SR Linux CLI with administrative permissions. The default password for the `linuxadmin` user is `NokiaSrl1!`, which can be changed; see Configuring the password for the linuxadmin user. Nokia recommends that you change this default user and password as soon as possible. Other Linux users can be added with the **useradd** command in the underlying Linux OS.

Linux users with a UID less than 1500 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 SR Linux, such as authentication by a TACACS+ server group, a RADIUS server group, or password security for local users.

5.1.2 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.The default password for the `admin` user is `NokiaSrl1!`, which you can change using the SR Linux CLI; see Configuring authentication for local users. Nokia recommends that you change this default password as soon as possible.

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

5.1.3 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.

5.2 AAA functions

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

5.2.1 Authentication

For Linux users, 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+ or RADIUS server group, can be configured. 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.

SR Linux supports `yescrypt` encryption in addition to the argon2 (`ar2`) and SHA512 (`sha2`) algorithms. All local users who authenticate using plain text or hashed passwords, including the admin user, are subject to these encryptions. By default, `yescrypt` encryption is used to secure clear-text passwords. See Password hashing for local users.

For the `linuxadmin` user password, only the `sha2` and `yescrypt` encryption options are supported. If the `ar2` encryption is selected, the password is hashed using `yescrypt` instead.

5.2.1.1 Superuser attribute for local users

SR Linux default `admin` user, and local users have a configurable `superuser` parameter setting under `.system.aaa.authentication.user`. When configuring the `superuser` attribute, the local user gains elevated permissions and is included in the `/etc/sudoers` file. The local user can now use the **bash** plug-in and then `sudo` to run any command as the `root` user.

For information about configuring superuser attribute, see Configuring superuser attribute for local users.

5.2.1.2 Password security for local users

SR Linux supports password security features that apply to local users who authenticate using plain text or hashed passwords. The settings are system-wide configurations that apply to all locally configured users including, the admin and the `linuxadmin` user. The following password security features are supported:

- **password aging**

  Specifies the number of days after which a password expires.

  Note:

  – Users with aged out passwords are prompted to update their password at the next login to the CLI. (This prompt only appears on the CLI.)

  – SR Linux does not provide advanced warning to users that their password is about to expire.

  - Aged out passwords expire indefinitely. They are not controlled by the lockout policy described below.

- **password change on first login**

  Specifies whether local users must change their password on first login. The new password must match the defined password complexity rules.

- **password history**

  Specifies the number of previous user passwords SR Linux retains to prevent reuse of old passwords.

- **lockout policy**

  Specifies how many times a user can attempt a failed login within a defined period before they are locked out (for example: three attempts within 1 minute). The policy also defines whether the user is locked out indefinitely or only for a specified length of time.

- **Password complexity rules**

  Defines password requirements, including minimum and maximum length, required numbers of lowercase, uppercase, numeric, and special characters, restrictions on username inclusion, and limits on consecutive character repetition.

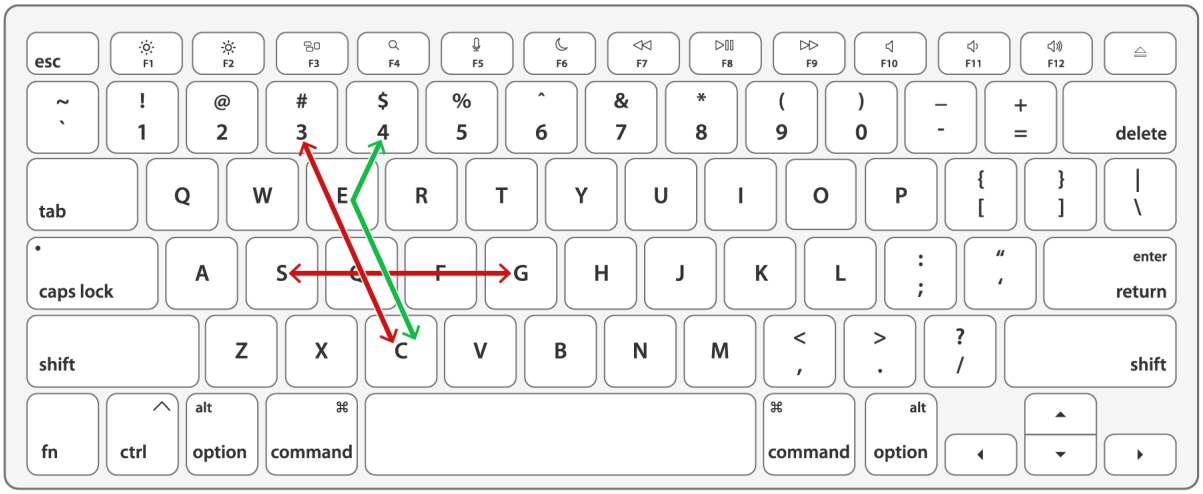
  The configuration of password complexity rules applies only to new passwords entered as clear text, not pre-existing or pre-hashed ones.

  Additionally, SR Linux restricts the use of sequential characters on a keyboard. You can configure a limit on the number of allowed numeric, uppercase, or lowercase characters that correspond to consecutive keys on U.S. English or Korean keyboard layout. These sequences are evaluated in all directions: left to right, right to left, down to up-right or up-left, or up to down-right or down-left.

  The number of sequential characters used in the password must not be equal to or exceed the configured value. Exceptions to this rule may be made for special characters, as specified in the password policy.

  The following figure shows the keyboard layout and key sequences for the password complexity rule.

  Figure 1: Keyboard layout and key sequences for password complexity rule



lw4571

For example, suppose the configured limit for the consecutive sequential character limit is set to four, the password like <4EDC> is considered valid because the movement from 4 to E is not a straight line on the keyboard. As a result, it only contains a sequence of three consecutive keys (<EDC>), which passes the validation. However, a password like 3EDC would be considered as a sequence of four keys and would fail the validation check. If the configured limit is set to three then even <4EDC > would fail the validation as the <EDC> part forms a sequence of three consecutive keys. For a configuration example, see Configuring password security for local users.

5.2.1.3 Password hashing for local users

Password hashing is applied to all locally configured users, including the admin and the `linuxadmin` user.

SR Linux supports `yescrypt` encryption in addition to the argon2 (`ar2`) and SHA512 (`sha2`) algorithms. You can choose to configure any of these supported methods.

The plain text user passwords are encrypted using the default hashing method `yescrypt`. When entering a hashed password value, the system verifies whether the entered password is encrypted using one of the supported algorithms. The system only commits changes if the password has been hashed using a supported algorithm.

When you change the password, the system encrypts the password using the configured hash method. See Configuring hash-method for local user passwords. The default hashing algorithm is `yescrypt`.

5.2.1.4 GLOME authentication and authorization

Generic Low Overhead Message Exchange (GLOME) login is a Google-originated protocol that provides secure authentication and authorization for low dependency environments.

GLOME provides a challenge-response mechanism similar to one-time authorization codes, presenting a solution to some of the challenges experienced with existing methods. Its key benefits include the following:

- **Zero-state**
  Session states do not need to be stored or managed on the server.
- **No clock dependency**
  GLOME works even when devices cannot maintain accurate time.
- **No predefined secret sharing**
  The operational overhead of secret distribution is eliminated.
- **Open Source**
  Full implementation details are publicly available on GitHub (google/glome).

GLOME operator workflow

The following section describes the general workflow for an operator connecting to the node:

1. **Login attempt** – You initiate a login attempt to the target node (for example, through SSH). The node responds with a GLOME challenge string, as shown in the following example.

```
GLOME: v2/gQgCsezpIjCW3o-PsI_Cpgtzw9_54d_cwIf_QnlopwU_/0_2/shell=linuxadmin/
```

2. **Challenge submission** – The operator takes the challenge code and inputs it into a tool or webpage to authenticate themselves and provide the challenge.
3. **User authentication** – The tool/webpage authenticates the user and provides the authorization code for this request, as shown in the following example.

```
rXusqSxUtM_ZjS8-p2V8dLtW1ENQEJNO2_827i5TvSA=
```

4. **Authorization code entry** – You enter the authorization code and gain access to the system, granted a valid response.

The following shows an example GLOME configuration.

Example: GLOME configuration

```
--{ candidate shared default }--[  ]--
A:admin@0_2# info with-context system aaa authentication login-method
    system {
        aaa {
            authentication {
                login-method {
                    console glome
                    remote glome
                    glome-key uNAQWmYejosPLSaeXFZJr0gJcn22q_K-IFvgBQgxJVA=
                }
            }
        }
    }
```

5.2.2 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.

All user types, including the default local user `admin`, are authorized through role-based control on all interfaces (CLI, gNMI, and JSON-RPC). However, the `linuxadmin` user is an exception and is granted write access to all commands in the command tree. Role-based access control supports service and CLI plug-in authorization. You can configure service authorization, which can limit the interface types for which the functions in a role are authorized. CLI plug-in authorization gives you control over how operator-provided plug-ins are loaded.

You can configure the SR Linux to use information from a TACACS+ or RADIUS server to assign roles to an authenticated user.

- In TACACS+ authorization, the priv-lvl value in the Authorization REPLY packet 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.
- RADIUS combines authentication and authorization. The RADIUS server authorizes the user by applying a profile based on username and password configurations. The profiles are configured using Vendor-Specific Attributes (VSAs) on the RADIUS server. The actions an authenticated user can perform depend on the user profiles. Profiles consist of a suite of commands that the user is allowed or not allowed to execute.

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

5.2.2.1 Superuser role attribute for local users

Local users have a configurable `superuser` role setting under **.system.aaa.authorization.role{}**. You can grant the `sudo` privileges to the individual users or groups, following the standard Linux semantics.

When a user has multiple roles with the `superuser` role setting enabled, the combined roles grant superuser privileges.

A superuser inherits the following behaviors:

- When the superuser is a member of multiple roles, the superuser is authorized to access all services specified in the assigned roles.
- When the superuser is not assigned any specific role, the superuser is authorized to access all services.

  Note: If a user has multiple roles and has no `superuser` role setting enabled, the user is authorized only for the services specified in the assigned roles.

- A superuser can load all CLI plug-ins, provided the superuser is associated with a role that supports CLI plug-in authorization.
- A superuser has an implicit role added that allows the user to write to the root directory (`/`).
- A superuser is added to the `/etc/sudoers` file and can use the **bash** plug-in and then `sudo` to run any command as the root user.

For information about configuring the superuser role attribute, see Configuring superuser role for local users.

5.2.3 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+ or RADIUS 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 local accounting, Configuring RADIUS accounting, and Configuring TACACS+ accounting for example configurations.

5.3 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 AAA via interaction with servers in a TACACS+ server group
- RADIUS – performs AAA via interaction with servers in a RADIUS server group

The AAA type that is configured is used for all AAA functions and cannot use different types for different functions. 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.

Each RADIUS or 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 authentication method is specified, or the server group is the last method in the list, then the authentication or accounting request is rejected.

The **health-check** command enables health check monitoring of the RADIUS and TACACS+ servers by sending requests at regular intervals, by default every 30 seconds. If a response is not received, the operational status of the server is changed to `down`. The operational status is changed to `up` when responses are received. Different health check monitoring requests are sent depending on the server type:

- RADIUS – An authentication request for the user `P1Z1X2C7S9Y9B008`*c*, where *c* is a random character, is sent and an Access-Reject response is expected.
- TACACS+ – A TCP connection to the server is opened and then closed.

5.3.1 Configuring an AAA server group

Procedure

You can configure authentication server groups for AAA functions. Configuring server groups provides a way to group the AAA server addresses. Grouping servers allows you to select a subset of the configured servers to use them for an AAA service. RADIUS and TACACS+ servers must be assigned to a network instance.

Example: Configure a local server group for AAA functions

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 SR Linux wait a maximum of 60 seconds for a local AAA function to complete.

```
--{ * candidate shared default }--[ ]--
# info with-context system aaa
    system {
        aaa {
            server-group LOCAL-GROUP {
                type local
                timeout 60
    }
```

Example: Configure a TACACS+ server group for AAA functions

The TACACS+ server group consists of three 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. The timeout period can be set for an individual server or for a server group.

SR Linux allows you to configure the source IP address of the packets sent from the router to the TACACS+ server. You can configure the **source-address** parameter as either an IPv4 address, IPv6 address, or FQDN under **server aaa server-group server tacacs** within a network instance. In the following example, the TACACS+ requests are sent from the configured IPv4 or IPv6 **source-address** in the **mgmt** network instance.

```
--{ * candidate shared default }--[  ]--
# info with-context system aaa
    system {
        aaa {
            server-group LOCAL-GROUP {
                type local
                timeout 60
            }
            server-group TACACS-GROUP {
                type tacacs
                timeout 30
                server 10.0.0.1 {
                    network-instance mgmt
                    tacacs {
                        secret-key $aes1$AWWxMe8CNxAZ/28=$tYLDBYE6boVHA1/4iKmpAg==
                        source-address 192.168.3.4
                    }
                }
                server 10.0.0.2 {
                    network-instance mgmt
                    tacacs {
                        secret-key $aes1$AWUxMTqSeZQVZG8=$lY73o0HeTwX8UMJiYiEnAQ==
                        source-address 192.168.3.4
                    }
                }
                server 10.0.0.3 {
                    network-instance mgmt
                    tacacs {
                        secret-key $aes1$AWUPUZthMjjSxG8=$No1xdN4/EE8Z1YCa7QBwCg==
                        source-address 192.168.3.4
                    }
                }
            }
        }
    }
```

Example: Configure RADIUS server group for AAA functions

The RADIUS server group consists of three 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. The timeout period can be set for an individual server or for a server group. The server address entries are tried in the order in which they are configured. By default, the RADIUS protocol uses port 1812 for authentication and authorization, and port 1813 for accounting.
SR Linux allows you to configure the source IP address of the packets sent from the router to the RADIUS server. You can configure the **source-address** parameter as either an IPv4 or IPv6 address under **server aaa server-group server radius** within a network instance. In the following example,

the RADIUS requests are sent from the configured IPv4 or IPv6 **source-address** in the **mgmt** network instance.

```
--{ * candidate shared default }--[  ]--
info with-context system aaa
    system {
        aaa {
            server-group RADIUS-GROUP {
                type radius
                timeout 30
                server 10.0.0.1 {
                    network-instance mgmt
                    radius {
                        auth-port 1812
                        acct-port 1813
                        secret-key $aes1$AWWDbQ87riUVxm8=$a56muaolKX9QVgZbE/cW3g==
                        source-address 192.168.3.4
                    }
                }
                server 10.0.0.2 {
                    network-instance mgmt
                    radius {
                        auth-port 1812
                        acct-port 1813
                        secret-key $aes1$AWXDSbXVCfU/IG8=$3fzuwfPb3CO1ZvvYBKFD1Q==
                        source-address 192.168.3.4
                    }
                }
                server 10.0.0.3 {
                    network-instance mgmt
                    radius {
                        auth-port 1812
                        acct-port 1813
                        secret-key $aes1$AWWhVcwDAb4Gt28=$jfPzjxRa9KC82ZV1VR2wSA==
                        source-address 192.168.3.4
                    }
                }
            }
        }
    }
```

5.4 Authentication for the linuxadmin user

The SR Linux `linuxadmin` user is installed by default in `/etc/passwd`. The `linuxadmin` user has access to `sudo` to root and can run the SR Linux CLI with admin permissions.

The default password for the `linuxadmin` user is `NokiaSrl1!`. You can change this password via the SR Linux CLI or any other supported interface.

5.4.1 Configuring the password for the linuxadmin user

Procedure

You can set a password for the `linuxadmin` user.

Note:
For the `linuxadmin` user password, only the `sha2` and `yescrypt` encryption options are supported. If the `ar2` encryption is selected, the password is hashed using `yescrypt` instead.

Example

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

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

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authentication linuxadmin-user
    system {
        aaa {
            authentication {
                linuxadmin-user {
                    password $y$j9T$2502d6515df892cf$NZ4.Dj0H7u7zsCvRar3v0Fx
BiKe5xKL5nmUX3W5UVP/
                }
            }
        }
    }
```

5.4.2 Recovering linuxadmin password

About this task

You can recover the password for the `linuxadmin` user.

WARNING: Ensure necessary backups are in place before proceeding, as this recovery procedure may cause data loss in the NOKIA-DATA labeled partition.

Procedure

Step 1. Reboot the device using console access.

Step 2. In the GRUB screen, press either the **ESC + 'e'** or **'e'** to edit boot options.

Step 3. Locate the line starting with **linux** or **linuxefi** and append **srl.formatovl** after **rw**.

Step 4. Press **Ctrl + X** or **F10** for the changes to take effect, and the device proceeds with booting.

Step 5. After the reboot, log in using the default login credentials.

- username: `linuxadmin`
- password: `NokiaSrl1!`

5.4.3 Restricting login access for local Linux users

Procedure

You can restrict login access for local Linux users by setting the parameter **disable-login** in the `system.aaa.authentication.local-linux-users` context.

- Set the parameter **disable-login** to `remote` to prevent local Linux users from accessing bash via remote login protocols like SSHv2.
- Set the parameter **disable-login** to `console` to prevent local Linux users from accessing bash via console.

Example: Restrict login access via remote login

The following example configuration restricts local Linux users from login access via remote.

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authentication local-linux-users disable-login
    system {
        aaa {
            authentication {
                local-linux-users {
                    disable-login [
                        remote
                    ]
                }
            }
        }
    }
```

Example: Restrict login access via console

The following example configuration restricts local Linux users from logging in via the console.

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authentication local-linux-users disable-login
    system {
        aaa {
            authentication {
                local-linux-users {
                    disable-login [
                        console
                    ]
                }
            }
        }
    }
```

5.4.4 Configuring fallback authentication for local Linux users

Procedure

SR Linux offers a fallback mechanism that allows remote access to devices, even when Linux user remote or console logins are disabled, and the `aaa_mgr` is unavailable or in a failed state. For information about restricting remote or console login, see Restricting login access for local Linux users.

You can enable the fallback authentication by setting the parameter, **allow-fallback** in the `system.aaa.authentication.local-linux-users` context.

To prevent accidental security misconfiguration, the **allow-fallback** parameter is set to `false` by default.

Enabling the **allow-fallback** parameter allows the local Linux users, restricted by the setting of the **disable-login** parameter in the `system.aaa.authentication.local-linux-users` context, to authenticate using their password, even if the `aaa_mgr` is unavailable.

Example: Configure fallback authentication

The following example configuration enables fallback authentication for local Linux users restricted from accessing bash via remote login protocols like SSHv2.

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authentication local-linux-users
    system {
        aaa {
            authentication {
                local-linux-users {
                    allow-fallback true
                    disable-login [
                        remote
                    ]
                }
            }
        }
    }
```

5.5 Authentication for local users

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

Note: For authentication functionality such as password lockout and the failed-login-attempts counter, the NTP server must be enabled and in synchronization.

5.5.1 Configuring authentication for local users

Procedure

By default, there is a single local user configured on the SR Linux, `admin`. The default password for the `admin` user is `NokiaSrl1!`. You can configure additional local users.

Note: When you change the password, the system encrypts the password as per the configured hash method. See Configuring hash-method for local user passwords. Default is argon2 (`ar2`).

Example: Configure a password for the SR Linux `admin` user

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

Example: Configure a password for a local user `srlinux`

```
--{ * 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 with context system aaa
```

```
    system {
        aaa {
            authentication {
                user srlinux {
                    password $ar2$KGSITqfJu5g=$iZZ6XKXtXOZOGMbeXO2ypg==
                }
            }
        }
    }
```

Example: Change an existing user's password

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: Authentication methods

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-GROUP`.

If the user cannot be authenticated through any of these methods, the authentication attempt is rejected.

```
--{ * candidate shared default }--[  ]--
# info with context system aaa
    system {
        aaa {
            authentication {
                authentication-method [
                    LOCAL-GROUP
                    TACACS-GROUP
                ]
            }
        }
    }
```

5.5.2 Configuring superuser attribute for local users

Procedure

To configure the superuser attribute, set the `superuser` parameter under **.system.aaa.authentication.user** to `true`. By default, the `superuser` attribute is disabled. This ensures backward compatibility with the previous releases.

Example: Configuring the superuser attribute for a local user

In this example, the `superuser` attribute setting is enabled for the `srl-test-user`user.

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authentication user srl-test-user superuser
    system {
        aaa {
            authentication {
```

```
                user srl-test-user {
                    superuser true
                }
            }
        }
    }
```

Example: Configuring the superuser attribute for SR Linux default admin user

The following example configures the `superuser` attribute setting for the SR Linux `admin` user. By default, the `superuser` attribute is enabled, ensuring backward compatibility and making the superuser status of the `admin` user more explicit.

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authentication admin-user superuser
    system {
        aaa {
            authentication {
                admin-user {
                    superuser true
                }
            }
        }
    }
```

5.5.3 Configuring password security for local users

Procedure

All local users who authenticate using plain text or hashed value passwords, including the admin user, are subject to password security and user lockout settings.

To configure password security and user lockout features, set the following parameters under **system aaa authentication**:

- **password aging** *<0 to 500>*
  Sets the number of days after which the user password expires and the user is prompted to update their password. The default of **0** sets the password to never expire.
- **password change-on-first-login [true | false]**
  When set to **true**, forces local users to change their password on the first login to the system. The default is **false**.
- **password history** *<0 to 20>*
  Specifies how many previous passwords SR Linux matches a new password against, such that the new password cannot be one of the previous *<0 to 20>* passwords. The default is **0**, which disables password history.
- **password complexity-rules**
  Sets the complexity rules that new passwords must match, using the following options:
  - **minimum-length** *<1 to 12>*
    Sets the minimum password length. The default is **1**, which applies no minimum.
  - **maximum-length** *<1 to 1023>*
    Sets the maximum password length. The default is **1023**.
  - **minimum-lowercase** *<0 to 10>*

Sets the minimum required lowercase characters. The default is **0**, which applies no minimum.

- **minimum-uppercase** *<0 to 10>*
  Sets the minimum required uppercase characters. The default is **0**, which applies no minimum.
- **minimum-numeric** *<0 to 10>*
  Sets the minimum required numeric characters. The default is **0**, which applies no minimum.
- **minimum-special-character** *<0 to 10>*
  Sets the minimum required number of special characters, which can include the following: (!"#$ %&'()*+,-./:;<=>?@[\]^_`{|}~"). The default is **0**, which applies no minimum.

  Note: Users must enter the double quote as \" and the backslash as \\.

- **allow-user-name** [**true** | **false**]
  Specifies whether the user can include their username as part of their password. If set to **false**, then SR Linux allows only the first three consecutive characters of the username in the user password. The default is **true**, which allows the password to contain the full username.
- **disallow-sequence-keys** *<0 |2 to 8>*
  Sets the maximum number of consecutive keys allowed in a sequence within a password. The password is rejected if it contains a sequence with this number or more consecutive keys. For example, if **disallow-sequence-keys** is set to four, a password can contain at most three consecutive keys in a sequence. The default is **0**, which means no limit is enforced.
- **repeated-characters** *<0 |2 to 8>*
  Sets the number of times a character can be repeated consecutively. If a password contains a sequence with more than the specified number of the same repeating character, the password is rejected. For example, if **repeated-characters** is set to 4, a password can contain at most four consecutive occurrences of the same character. The default is **0**, which means no limit is enforced.

- **password lockout-policy**
  Specifies the user lockout policy using the following parameters:
  - **attempts** *<0 to 64>*
    Sets the number of failed login attempts that trigger the lockout. The default is **0**, which allows unlimited failed login attempts.
  - **time** *<0 to 1440>*
    Sets the time period in minutes within which the failed login attempts must occur to trigger the lockout. The timer starts at the first failure. The default is **1** minute.
  - **lockout** *<0 to 1440>*
    Sets the time period in minutes during which the user account remains locked out. A value of **0** means that the user account is locked out indefinitely. The default is **15** minutes.
- **require-ntp-sync [true | false]**
  When set to **false**, this option forces the NTP synchronization check to be ignored and use only the system clock. The default is **true**. You can configure this setting in circumstances where the configured NTP servers are unreachable or unreliable. The configurable setting to ignore the NTP sync status is included in the password change-on-first-login (`.system.aaa.authentication.password.change-on-first-login`), password aging (`.system.aaa.authentication.password.aging`), and lockout policy mechanism (`.system.aaa.authentication.password.lockout-policy`).

When included in the above features, the command **require-ntp-sync** disables NTP synchronization check. However, this command does not prevent the operator from syncing the system clock to NTP. It simply removes the dependency on NTP synchronization for those features.

Example: Configure password security and user lockout

The following example sets the following password security and user lockout settings:

- User passwords expire after 365 days.
- Users must change their password on first login.
- Users cannot reuse any of their 10 previous passwords.
- Forces the NTP synchronization check to be ignored and use only the system clock.
- Passwords must contain:
  - a minimum of eight and a maximum of 16 characters
  - one each of lowercase, uppercase, numeric, and special characters
  - only up to three consecutive keys in sequence
  - only up to three consecutive occurrences of the same character
- Usernames are not allowed in the passwords.
- After five failed attempts within 2 minutes, users are locked out indefinitely.

```
--{ * candidate shared default }--[ ]--
# info with-context system aaa authentication password
    system {
        aaa {
            authentication {
                password {
                    aging 365
                    change-on-first-login true
                    history 10
                    require-ntp-sync false
                    complexity-rules {
                        minimum-length 8
                        maximum-length 16
                        minimum-lowercase 1
                        minimum-uppercase 1
                        minimum-numeric 1
                        minimum-special-character 1
                        allow-username false
                        disallow-sequence-keys 4
                        repeated-characters 4
                    }
                }
            }
        }
    }
```

5.5.4 Configuring hash-method for local user passwords

Procedure

For local-user and admin-user passwords, you can choose the password hashing method by setting the `hash-method` parameter under **.system.aaa.authentication.password**.

By default, the `hash-method` is set to `yescrypt`.

Example: Configuring the `yescrypt` hashing algorithm

```
--{ * candidate shared default }--[ ]--
# info with context system aaa authentication password hash-method
    system {
        aaa {
            authentication {
                password {
                    hash-method yescrypt
                }
            }
        }
    }
```

Example: Configuring the `ar2` hashing algorithm

```
--{ * candidate shared default }--[ ]--
# info with context system aaa authentication password hash-method
    system {
        aaa {
            authentication {
                password {
                    hash-method ar2
                }
            }
        }
    }
```

Example: Configuring the `sha2` hashing algorithm

```
--{ * candidate shared default }--[ ]--
# info with context system aaa authentication password hash-method
    system {
        aaa {
            authentication {
                password {
                    hash-method sha2
                }
            }
        }
    }
```

5.5.5 Clearing locked-out local users

Procedure

To display locked out users, use the **info from state system aaa authentication** command, which displays a `lockout` state of `active true` when a user is locked out.

To clear a local user's lockout state, use the following tools command:

tools system aaa authentication user <*username*> **unlock**

Example: Display user lockout state

```
--{ * candidate shared default }--[ ]--
#info with context from state system aaa authentication
    system {
        aaa {
            authentication {
                exit-on-reject false
                idle-timeout 600
                authentication-method [
                    local
                ]
                user srl-test-1 {
                    password $bt$Vbf0Zgg8MGP=$ihHklTQHi/+3VwVD04Z8gh==
                    failed-login-attempts 10
                    last-failed-login 2022-07-29T22:35:23.801Z
                    password-change-required true
                    role [
                        test
                    ]
                    lockout {
                        active true
                        start 2022-07-29T22:35:23.801Z
                        end "14 minutes from now"
                    }
                }
                user srl-test-2 {
                    password $bt2$jBZBL6WAuTr=$eXYWpivE5z10YxBIZ06hjQ==
                    password-change-required true
                    role [
                        test
                    ]
                }
```

Example: Clear user lockout state

```
--{ * candidate shared default }--[ ]--
#tools system aaa authentication user srl-test-1 unlock
/system/aaa/authentication/user[username=srl-test-1]:
    Unlocked user srl-test-1
```

5.6 Authorization using role-based access control

Authorization via role based access control is performed for all user types including the default local user admin (`admin`), with the exception of the `linuxadmin`/`root` users, which are permitted write access to all

commands in the command tree. Users can be configured with a set of one or more roles that define 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 and CLI plug-in authorization.

- Service authorization allows you to limit the actions a user is authorized to perform to specific access types such as CLI and gNMI.
- CLI plug-in authorization gives you control over how operator-provided plug-ins are loaded. You can choose to load them from the global plug-in directory or from the user's home directory. This feature also allows you to control the execution of CLI plug-ins and the queries they make against the data model. Additionally, you can manage the list of CLI commands that are allowed or not allowed to execute.

5.6.1 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 forward slash "`/`" 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 configuration is equivalent to a rule with the forward slash "`/`" as the path-reference and `deny` as the action.

When no roles are configured, the `admin` retains its superuser privileges. This ensures backward compatibility with previous releases that only allowed role configuration for all users except for the `admin` and `linuxadmin` users.

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 the forward slash "`/`" 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 must 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\""`

The roles control the service and CLI plug-in authorization .

5.6.1.1 Configuring a role

Procedure

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: Configure a role with multiple rules

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 on 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 with context system aaa
    system {
        aaa {
            authorization {
                role testrole {
                }
            }
        }
    }
```

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

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

Example: Configure a role with view-only access permission

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 with context system aaa
    system {
        aaa {
            authorization {
                role acl_app {
                }
            }
        }
```

```
    }
```

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

5.6.1.2 Configuring superuser role for local users

Procedure

To configure the superuser role attribute, set the `superuser` role parameter under **.system.aaa.authorization.role{}** to `true`. By default, the `superuser` role parameter is disabled. This ensures backward compatibility with the previous releases.

Example: Configuring the superuser role attribute

In this example, the `test_role` is configured with the `superuser` role setting..

```
--{ * candidate shared default }--[  ]--
# system aaa authorization role test_role
    system {
        aaa {
            authorization {
                role test_role {
                    superuser true
                }
            }
        }
    }
```

Example: Assigning the superuser role to a local user

In the following example, the previously configured superuser role (`test_role`) is assigned to the `srl-test-user` user.

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authentication user srl-test-user
    system {
        aaa {
            authentication {
                user srl-test-user {
                    role [
                        test_role
                    ]
                }
            }
        }
    }
```

5.6.2 Assigning roles to users

Procedure

You can assign a user one or more roles. The rules configured in the user's role 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.

When it comes to CLI plug-in authorization, any user who has the necessary role to load the plug-in can do so. The system follows an additive and permissive approach.

Example

In the following example, the default local user `admin` and user `testuser` are assigned the role `testrole`.

After the `testuser` 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 assignment authorizes the `testuser` user to configure everything in the system except for the network-instance `red`.

```
--{ * candidate shared default }--[  ]--
# info with context system aaa
    system {
        aaa {
            authentication {
                idle-timeout 7200
                authentication-method [
                    local
                ]
                admin-user {
                    role [
                        testrole
                    ]
                }
```

```
                linuxadmin-user {
                    password $6$skWKUdwWyM/$muyCaWpqJyax
Yrs0J6nzb8tiM63W1HGKnsSOrb4JA7PiexF1q89PdualYQtyfl/q1qJJ9qyVE0CeTF0Csu87a/
                }
                user testuser {
                    role [
                        testrole
                    ]
                }
            }
        }
    }
```

5.6.3 Authorization using a TACACS+ server

You can configure SR Linux to use a TACACS+ server to provide authorization for role-based access control. When TACACS+ authorization is used, the actions an authenticated user can perform can be configured with privilege-level mapping, services, and vendor-specific attributes (VSAs, or both). If neither are configured, the authenticated user is authorized as an admin user.

TACACS+ authorization with privilege-level mapping

TACACS+ authorization with privilege-level mapping works as follows:

1. After a user is authenticated, SR Linux sends the TACACS+ server an authorization request message for the shell service.
2. The TACACS+ server returns an authorization reply message that includes the **priv-lvl** value configured for the user.
3. Locally configured roles are mapped to the **priv-lvl** value.
4. Users can perform actions specified in roles with a **priv-lvl** value equal to or lower than the priv-lvl value returned by the TACACS+ server.

TACACS+ authorization with services and VSAs

TACACS+ authorization with services and VSAs works as follows:

1. After a user is authenticated, SR Linux sends the TACACS+ server authorization request messages for the services that are configured on the router. A temporary role with no access permissions is created for the user.

   If privilege-level mapping is also configured, SR Linux sends the TACACS+ server an authorization request message for the shell service.
2. The TACACS+ server returns an authorization reply message with services and VSAs configured for the user.

   If privilege-level mapping is also configured, the TACACS+ server returns an authorization reply message that includes the **priv-lvl** value configured for the user.
3. The temporary role is populated with the service VSAs that are received:
   - Invalid VSA values are rejected and authentication fails.
   - Unknown mandatory VSAs are rejected and authentication fails.
   - Unknown optional VSAs are discarded and authentication succeeds.

   If privilege-level mapping is also configured, locally configured roles are mapped to the **priv-lvl** value.

4. Users can perform actions specified in the temporary role.

5.6.3.1 Configuring TACACS+ authorization with privilege-level mapping

Procedure

To configure TACACS+ authorization using privilege levels, you configure local roles that specify the privilege-level mapping for each role subject to authorization by a TACACS+ server.

Example: Enable authorization with privilege levels for a TACACS+ server group

The following configuration enables privilege-level authorization for the **TACACS-GROUP** server group:

```
--{ * candidate shared default }--[  ]--
# info with-context system aaa server-group TACACS-GROUP
    system {
        aaa {
            server-group TACACS-GROUP {
                type tacacs
                tacacs {
                    priv-lvl-authorization true
                }
            }
        }
    }
```

Example: Configure roles with read access

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 operational states, and the **network-instance oper-state** role grants read access for all network-instance operational states.

```
--{ * candidate shared default }--[  ]--
# info with-context system aaa server-group TACACS-GROUP
    system {
        aaa {
            server-group TACACS-GROUP {
                type tacacs
                tacacs {
                    priv-lvl-authorization true
                }
            }
        }
    }
```

Example: Enable privilege-level mapping for the configured roles

The following configuration specifies the privilege-level mapping for both roles.

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 operational states and all network-instance operational 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 network-instance operational states, but has no access to anything else in the system.

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

```
# info with-context system aaa authorization
    system {
        aaa {
            authorization {
                role "interface oper-state" {
                    tacacs {
                        priv-lvl 14
                    }
                }
                role "network-instance oper-state" {
                    tacacs {
                        priv-lvl 13
                    }
                }
            }
        }
    }
```

Example: Configure service authorization for a role

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-GROUP**, 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 with-context 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**.

5.6.3.2 TACACS+ services and VSAs

SR Linux supports several services with VSAs for user roles and role-based access control. Administrators can optionally configure the service and VSAs for each user on a TACACS+ server instead of configuring access controls locally.

Services and VSAs are available for the following access controls:

- The **nokia-srl-authorization-role** service and VSAs for user roles that correspond to **system aaa authorization role** commands.
- The **nokia-srl-authorization-role** service and VSAs for access to services that correspond to **system aaa authorization role service** commands.
- The **nokia-srl-authorization-role-cli** service and VSAs for access to CLI commands that correspond to **system aaa authorization role cli** commands.
- The **nokia-srl-authorization-role-netconf** service and VSAs for NETCONF RPCs that correspond to the **system aaa authorization role netconf allowed-operations** commands.
- The **nokia-srl-configuration-role** service and VSAs for path permissions that correspond to the **system configuration role** commands.

Roles are applied in the following precedence order to create the most restrictive default values. When a step matches, no further profiles are applied.

- For **nokia-srl-authorization-role**, **nokia-srl-authorization-role-cli**, and **nokia-srl-authorization-role-netconf** services:
  1. a temporary role with no permissions
  2. VSAs that are received
  3. privilege-level mapping
- For the **nokia-srl-configuration-role** service:
  1. a temporary role with no permissions
  2. VSAs that are received

All service names, VSA names, and values must be entered in lowercase in the TACACS+ server configuration.

Table 5: VSAs for user roles

Service name	VSA name	Description	Values
nokia-srl-authorization-role	superuser	Indicates if users with this role are given superuser access	true – gives the user superuser access false – denies the user superuser access

Table 6: VSAs for access to services

Service name	VSA name	Description	Values
nokia-srl-authorization-role	services-cli	The CLI service	true – permits access

Service name	VSA name	Description	Values
			false – denies access
nokia-srl-authorization-role	services-ftp	The FTP service	true – permits access false – denies access
nokia-srl-authorization-role	services-gnmi	The gNMI service	true – permits access false – denies access
nokia-srl-authorization-role	services-gnoi	The gNOI service	true – permits access false – denies access
nokia-srl-authorization-role	services-gnsi	The gNSI service	true – permits access false – denies access
nokia-srl-authorization-role	services-gribi	The gRIBI service	true – permits access false – denies access
nokia-srl-authorization-role	services-grpc-reflection	The gRPC reflection service	true – permits access false – denies access
nokia-srl-authorization-role	services-json-rpc	The JSON RPC service	true – permits access false – denies access
nokia-srl-authorization-role	services-netconf	The NETCONF service	true – permits access false – denies access
nokia-srl-authorization-role	services-p4rt	The P4Runtime service	true – permits access false – denies access

Table 7: VSAs for access to CLI commands

Service name	VSA name	Description	Values
nokia-srl-authorization-role-cli	local-global-plugin	Specifies whether the CLI loads plugins from the global plugin directory (from `/etc/opt/srlinux/cli/plugins/`)	true – loads global plugins false – does not load global plugins
nokia-srl-authorization-role-cli	load-user-plugin	Specifies whether the CLI loads plugins from the user's home directory (from `~/cli/plugins/`)	true – loads local plugins false – does not load local plugins
nokia-srl-authorization-role-cli	allow-command*N*	The allowed command number *N*. More than	A string up to 100 characters. The

Service name	VSA name	Description	Values
		one allowed command can be created with unique numbers. Exits when the longest match is found. Commands should be sequenced from most explicit to least explicit.	syntax is the same as the **system aaa authorization role operations cli allow-command-list** command syntax. Example: `allow-command1 = "ping"`
nokia-srl-authorization-role-cli	deny-command*N*	The denied command entry number *N*. More than one denied command can be created with unique numbers. Exits when the longest match is found. Commands should be sequenced from most explicit to least explicit.	A string up to 100 characters. The syntax is the same as the s**ystem aaa authorization role operations cli deny-command-list** command syntax. Example: `deny-command1 = "bash"`

The following CLI command authorization usage guidelines apply:

- The range of entry numbers is 1 to 9999.
- Leading zeros before the action or entry number are invalid, for example **allow-command01**.
- The entries are sorted by the index into a leaf-list and the longest match is evaluated, the same way as in the local role configuration.
- These commands can be in any order in the TACACS+ configuration file.

Table 8: VSAs for NETCONF RPCs

Service name	VSA name	Description	Values
nokia-srl-authorization-role-netconf	allowed-operations-action	Allow the NETCONF <action> RPC	true – permits the RPC false –denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-cancel-commit	Allow the NETCONF <cancel-commit> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-close-session	Allow the NETCONF <close-session> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-commit	Allow the NETCONF <commit> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-copy-config	Allow the NETCONF <copy-config> RPC	true – permits the RPC false – denies the RPC

Service name	VSA name	Description	Values
nokia-srl-authorization-role-netconf	allowed-operations-delete-config	Allow the NETCONF <delete-config> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-discard-changes	Allow the NETCONF <discard-changes> RPC	true – permits the RPC false –denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-edit-config	Allow the NETCONF <edit-config> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-edit-data	Allow the NETCONF <edit-data> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-get	Allow the NETCONF <get> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-get-config	Allow the NETCONF <get-config> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-get-data	Allow the NETCONF <get-data> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-get-schema	Allow the NETCONF <get-schema> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-kill-session	Allow the NETCONF <kill-session> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-lock	Allow the NETCONF <lock> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-unlock	Allow the NETCONF <unlock> RPC	true – permits the RPC false – denies the RPC
nokia-srl-authorization-role-netconf	allowed-operations-validate	Allow the NETCONF <validate> RPC	true – permits the RPC false – denies the RPC

Table 9: VSAs for path permissions

Service name	VSA name	Description	Values
nokia-srl-configuration-role	rule*N*	The rule number *N*. More than one rule can be created with unique numbers. Exits when the longest match is found. Rules should be	A string up to 255 characters. The syntax is the same as the **system configuration role rule** command syntax.

Service name	VSA name	Description	Values
		sequenced from most explicit to least explicit.	Example: `rule1 = "interface * oper-state"`
nokia-srl-authorization-role-cli	action*N*	The action number *N* associated with rule number *N*	deny – denies paths matching the rule read – permits read access to paths matching the rule write – permits write access to paths matching the rule Example: `action1 = read`

The following path permission usage guidelines apply:

- The range of entry numbers is 1 to 9999.
- Leading zeros before the action or entry number are invalid. For example, **allow-command01**.
- The rules are sorted by their entry number and evaluated in numerical order, the same way as evaluation in local rules.
- One rule and one corresponding action index must be present, but they can be in any order in the TACACS+ configuration file. A maximum of 126 rules and 126 actions can be created.

5.6.3.3 Configuring TACACS+ authorization with VSAs

Command authorization can be configured with TACACS+ VSAs instead of local roles or privilege-level mapping. This allows authorization to be controlled centrally on the TACACS+ server by an administrator, instead of being configured on the router. The command authorization role is sent in VSAs from the TACACS+ server to the router after the user authenticates. The command authorization role is then installed locally on the router in a temporary profile for each user.

To configure TACACS+ authorization using services and VSAs, you can enable the services needed for your authorization requirements on SR Linux and configure each user with the services and VSAs on the TACACS+ server.

Example: SR Linux configuration

The following example shows these configurations:

- On the router:
  - All users are authenticated via TACACS+.
  - The **nokia-srl-authorization-role**, **nokia-srl-authorization-role-cli**, and **nokia-srl-configuration-role** services are enabled.
- On the TACACS+ server: Command authorization is configured.

```
--{ * candidate shared default }--[  ]--
# info with-context system aaa server-group TACACS-GROUP
```

```
    system {
        aaa {
            server-group TACACS-GROUP {
                    tacacs {
                        service-request {
                            nokia-srl-authorization-role true
                            nokia-srl-authorization-role-cli true
                            nokia-srl-configuration-role true
                        }
                    }
                }
            }
        }
```

Example: TACACS+ server configuration for a permissive role

The user1 profile uses VSAs to permit the user access to the CLI and all CLI commands, with a role that has access to all configuration and state.

```
user = user1 {
    service = nokia-srl-authorization-role {
        services-cli = true
    }
    service = nokia-srl-configuration-role {
        rule1 = /
        action1 = write
    }
}
```

Example: TACACS+ server configuration for a restrictive role

The user2 profile uses VSAs to permit the user access to the CLI and all CLI commands, with a restrictive role that cannot read or write **system aaa** configuration or state.

```
user = user2 {
    service = nokia-srl-authorization-role {
        services-cli = true
    }
    service = nokia-srl-configuration-role {
        rule1 = /
        action1 = write
        rule2 = "system aaa"
        action2 = deny
    }
}
```

Example: TACACS+ server configuration for an operations role

The user3 profile uses VSAs to permit the user access to the CLI, and restricts access to some CLI commands, with a restrictive role that can only display interface state.

```
user = user3 {
    service = nokia-srl-authorization-role {
        services-cli = true
    }
    service = nokia-srl-authorization-role-cli {
        deny-command1 = bash
        deny-command2 = "enter candidate"
        deny-command3 = save
    }
    service = nokia-srl-configuration-role {
```

```
        rule1 = /
        action1 = deny
        rule2 = "interface * oper-state"
        action2 = read
    }
}
```

5.6.4 Authorization using a RADIUS server

RADIUS combines authentication and authorization. The RADIUS server authorizes the user by applying a profile based on username and password configurations. The profiles are configured using VSAs on the RADIUS server. The actions an authenticated user can perform depend on the user profiles. Profiles consist of a suite of commands that the user is allowed or not allowed to execute.

SR Linux requires the `Timetra-SRL-Auth-User-Role <string>` VSA, which is mapped to authorization roles configured on the router. The RADIUS dictionary file `dictionary-freeradius.txt`, in the `support` directory of the SR Linux software distribution, includes the supported VSAs.

The Nokia-defined attributes are encapsulated in a RADIUS vendor-specific attribute with the vendor ID field set to 6527.

RADIUS authorization for SR Linux users works as follows:

1. The SR Linux sends the username and password to the RADIUS server.
2. If the username and password are valid, the RADIUS server returns the roles for the user in the `Timetra-SRL-Auth-User-Role <string>` VSA. All roles in the VSA are validated, and the user is disconnected if any role sent in the VSA does not exist locally.

   If the username and password are not recognized, access is denied and passed on to the next configured authentication method. If no other authentication method is configured, or the RADIUS server group is the last method in the list, then the authentication/authorization request is rejected.

   Note: The user profiles downloaded from the RADIUS server are stored on the SR Linux only for the user session. These profiles are considered temporary configurations and are not saved when the user session terminates.

3. Users authenticated with RADIUS and a user profile use RADIUS authorization, and not local authorization. When a user issues a command, SR Linux compares the command and the user information against the information present in the downloaded RADIUS user profile. The profile provides details about the command that the user is authorized and not authorized to execute. The user can execute only the authorized commands. If a user profile is not received, the user is disconnected.

5.6.4.1 Configuring RADIUS authorization

Procedure

To configure RADIUS authorization for SR Linux users, you enable authorization for the RADIUS server group and configure the UDP port number. By default, the RADIUS protocol uses port 1812 for authentication and authorization.

Example

The following example configures services authorization for a role that uses RADIUS authorization.

```
--{ * candidate shared default }--[  ]--
# info with context system aaa
    system {
        aaa {
            authorization {
                role role_1 {
                    services [
                        cli
                        gnmi
                        ftp
                    ]
                }
            }
        }
    }
```

5.6.5 Service authorization

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.

You can configure the service and CLI plug-in authorization all user types including the default local user admin (`admin`), with the exception of the `linuxadmin`/`root` users, which are permitted write access to all commands in the command tree.

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 statistics only via the gNMI interface.

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
- `p4rt` – access to the system via P4Runtime RPCs

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.

5.6.5.1 Configuring service authorization

Procedure

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: Create a role

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

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

Example: Assign the role to a user

The following example assigns `read-oper-state` role to the default local user `admin`. Additionally, it creates a user `gnmiuser` that is assigned the `read-oper-state` role when authenticated.

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authentication
    system {
        aaa {
            authentication {
                authentication-method [
                    local
                ]
                admin-user {
                    role [
                        read-oper-state
                    ]
                }
                user gnmiuser {
                    role [
                        read-oper-state
                    ]
                }
```

```
            }
        }
    }
```

Example: Configure service authorization for the role

The following example configures service authorization so that users assigned the `read-oper-state` role, such as `admin`/`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 with context system aaa authorization
    system {
        aaa {
            authorization {
                role read-oper-state {
                    services [
                        gnmi
                    ]
                }
            }
        }
    }
```

5.6.6 CLI plug-in authorization for local users

CLI plug-in authorization gives you control over how operator-provided plug-ins are loaded. You can choose to load them from the global plug-in directory (`/etc/opt/srlinux/cli/plugins/`) or from the user's home directory (`~/cli/plugins/`). This feature also allows you to control the execution of CLI plug-ins and the queries they make against the data model. Additionally, you can manage the list of CLI commands that are allowed or not allowed to execute.

You can configure the CLI plug-in authorization for all user types including the default local user admin (`admin`), with the exception of the `linuxadmin`/`root` users, which are permitted write access to all commands in the command tree.

To configure CLI plug-in authorization, you assign roles to users that authorize them to load plug-ins from either the global or user plug-ins directory. Additionally, you can specify the list of CLI commands that users assigned to the role are authorized to execute.

For example, you can define a role that grants permission to load plug-ins only from users home directory. When you assign this role to a user, the user can only load plug-ins from the home directory.

After a user is authenticated, the system checks whether the user is assigned a role that allows loading of plug-ins from the intended location and has permission to execute the intended CLI commands. If the authorization is successful, the user can perform the functionality as defined in the role. If the authorization fails, the user is denied access to the functionality defined in the role.

To configure the CLI plug-in authorization, set the following parameters under the following command:

```
system aaa authorization role role-name cli
```

:

- **allow-command-list**

Specify the list of CLI commands allowed to execute.

- **deny-command-list**
  Specify the list of CLI commands not allowed to execute.
- **load-global-plugins** [**true** | **false**]
  Specifies whether CLI should load plug-ins from global plug-in directory (`/etc/opt/srlinux/cli/plugins/`). If set to **false**, no plug-ins are loaded from the global plug-in directory. The default is **true**, which allows plug-ins to be loaded from the global plug-in directory.
- **load-user-plugins** [**true** | **false**]
  Specifies whether CLI should load plug-ins from user's home directory (`~/cli/plugins/`). If set to **false**, no plug-ins are loaded from the user's home directory. The default is **true**, which allows plug-ins to be loaded from the user's home directory.

5.6.6.1 Configuring CLI plug-in authorization

Procedure

To configure CLI plug-in authorization, you assign roles to users that authorize them to load plug-ins from either the global or user plug-ins directory. Additionally, you can specify the list of CLI commands that users assigned to the role are authorized to execute.

Example: Configure CLI plug-in authorization for the role

The following example configures CLI authorization so that users assigned to the `loadglobal-plugins` role, such as `cliuser`, can load only the global plug-ins and execute only the **info** and **tools** command without having access to the **bash** commands.

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authorization role loadglobal-plugins cli
    system {
        aaa {
            authorization {
                role loadglobal-plugins {
                    cli {
                        load-global-plugins true
                        load-user-plugins false
                        deny-command-list [
                            bash
                        ]
                        allow-command-list [
                            info
                            tools
                        ]
                    }
                }
            }
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authentication user cliuser
    system {
        aaa {
            authentication {
                user cliuser {
                    role [
```

```
                        loadglobal-plugins
                    ]
                }
            }
        }
    }
```

5.7 Accounting configuration

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

5.7.1 Configuring local accounting

To configure local accounting for SR Linux users, enable accounting for the local server group.

SR Linux can generate an accounting record when a command is stopped, or when it is started and stopped.

The following example configures accounting records to be sent to the **LOCAL-GROUP** server group.

```
--{ * candidate shared default }--[  ]--
# info with-context system aaa
    system {
        aaa {
            accounting {
                accounting-method [
                    LOCAL-GROUP
                ]
                event commands {
                    record start-stop
                }
            }
        }
    }
```

The following example displays command accounting in the `/var/log/srlinux/debug/sr_aaa_mgr.log` file.

```
2025-10-06T12:27:42.355946-04:00 node-2 local6|INFO sr_aaa_mgr: accounting|2160|2868|00066|
I: admin   pts/8   default 10.86.108.138   start   task_id=7       timezone=EDT    service=
srlinux-cli       priv-lvl=15     cmd=info
2025-10-06T12:27:42.362663-04:00 node-2 local6|INFO sr_aaa_mgr: accounting|2160|2868|00067|
I: admin   pts/8   default 10.86.108.138   stop    task_id=7       timezone=EDT    service=
srlinux-cli       priv-lvl=15     cmd=info
```

5.7.2 Configuring TACACS+ accounting

Procedure

To configure TACACS+ accounting for SR Linux users, enable accounting for the TACACS+ server group.

SR Linux can generate an accounting record when a command is stopped, or when it is started and stopped.

Example

The following example configures accounting records to be sent to the **TACACS-GROUP** server group.

```
--{ * candidate shared default }--[  ]--
# info with-context system aaa
    system {
        aaa {
            accounting {
                accounting-method [
                    TACACS-GROUP
                ]
                event commands {
                    record start-stop
                }
            }
        }
    }
```

The following example displays command accounting in the TACACS+ server log file.

```
Sep 16 12:38:13 127.0.0.1       admin   unknown 192.168.148.8   start   task_id=15
 timezone=UTC    service=shell       priv-lvl=15     cmd=info
Sep 16 12:38:13 127.0.0.1       admin   unknown 192.168.148.8   stop    task_id=15
 timezone=UTC    service=shell       priv-lvl=15     cmd=info
```

5.7.3 Configuring RADIUS accounting

Procedure

To configure RADIUS accounting for SR Linux users, enable accounting for the RADIUS server group and optionally configure the accounting port.

SR Linux generates an accounting record when a command is stopped, or when it is started and stopped. The accounting records are sent to the RADIUS server using UDP packets on port 1813 in the `Timetra-SRL-Cmd` VSA.

Example

The following example configures accounting records to be sent to the **RADIUS-GROUP** server group.

```
--{ * candidate shared default }--[  ]--
# info with-context system aaa
system {
    aaa {
        accounting {
            accounting-method [
                RADIUS-GROUP
            ]
            event commands {
                record start-stop
            }
         }
     }
 }
```

Example

The following example is an accounting start record sent to a RADIUS server.

```
Acct-Status-Type (40) = Start
NAS-IP-Address (4) = "192.168.146.101"
User-Name (1) = "user151"
Acct-Session-Id (44) = "000026992026-01-23T15:35:57.279Z"
Calling-Station-Id (31) = "192.168.146.124"
NAS-Port-Type (61) = Virtual
Timetra-SRL-Cmd (242.26.6527.2) = "show system aaa authentication session 33"
```

5.7.4 Configuring TACACS+ accounting

Procedure

To configure TACACS+ accounting for SR Linux users, enable accounting for the TACACS+ server group.

SR Linux can generate an accounting record when a command is stopped, or when it is started and stopped.

Example

The following example configures accounting records to be sent to the **TACACS-GROUP** server group.

```
--{ * candidate shared default }--[  ]--
# info with-context system aaa
    system {
        aaa {
            accounting {
                accounting-method [
                    TACACS-GROUP
                ]
                event commands {
                    record start-stop
                }
            }
        }
    }
```

The following example displays command accounting in the TACACS+ server log file.

```
Sep 16 12:38:13 127.0.0.1       admin   unknown 192.168.148.8   start   task_id=15
 timezone=UTC    service=shell       priv-lvl=15     cmd=info
Sep 16 12:38:13 127.0.0.1       admin   unknown 192.168.148.8   stop    task_id=15
 timezone=UTC    service=shell       priv-lvl=15     cmd=info
```

5.8 Displaying user session information

Procedure

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

Example

```
# show system aaa authentication session
+----+----------+---------+--------------+-------+-----+------------+----------+
| ID |   User   | Service | Authentica   | Priv- | TTY |  Remote    | Login    |
|    |   name   |   name  |   cation     |  lvl  |     |   host     | time     |
|    |          |         |   method     |       |     |            |          |
+====+==========+=========+==============+=======+=====+============+==========+
| 30 | admin    | sshd    | LOCAL-GROUP  |       | ssh | 172.18.0.1 | 2023-11- |
|    |          |         |              |       |     |            | 02T04:04 |
|    |          |         |              |       |     |            | :04.040Z |
| 31 | bob      | sshd    | TACACS-GROUP |  15   | ssh | 172.18.0.1 | 2023-11- |
|    |          |         |              |       |     |            | 02T04:05 |
|    |          |         |              |       |     |            | :36.854Z |
| 32 | user151* | sshd    | RADIUS-GROUP |       | ssh | 172.18.0.1 | 2023-11- |
|    |          |         |              |       |     |            | 02T04:05 |
|    |          |         |              |       |     |            | :36.854Z |
+----+----------+---------+--------------+-------+-----+------------+----------+
```

5.9 Disconnecting user sessions

Procedure

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 | Authentica   | Priv- | TTY |  Remote    | Login    |
|    |   name   |   name  |   cation     |  lvl  |     |   host     | time     |
|    |          |         |   method     |       |     |            |          |
+====+==========+=========+==============+=======+=====+============+==========+
| 30 | admin    | sshd    | LOCAL-GROUP  |       | ssh | 172.18.0.1 | 2023-11- |
|    |          |         |              |       |     |            | 02T04:04 |
|    |          |         |              |       |     |            | :04.040Z |
| 31 | bob      | sshd    | TACACS-GROUP |  15   | ssh | 172.18.0.1 | 2023-11- |
|    |          |         |              |       |     |            | 02T04:05 |
|    |          |         |              |       |     |            | :36.854Z |
| 32 | user151* | sshd    | RADIUS-GROUP |       | ssh | 172.18.0.1 | 2023-11- |
|    |          |         |              |       |     |            | 02T04:05 |
|    |          |         |              |       |     |            | :36.854Z |
+----+----------+---------+--------------+-------+-----+------------+----------+

# tools system disconnect session-id 31
Terminating cli session 31 owned by user 'bob' logged from ssh
/system/aaa/authentication/session[id=31]:
Disconnecting aaa cli session(s): 31
```

5.10 Configuring the idle-timeout period for user sessions

Procedure

You can configure the idle-timeout period for user sessions, which disconnects a user session after a specified period of inactivity. By default, user sessions are disconnected after 10 minutes of inactivity.

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

When a user session is 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 period so that SR Linux user sessions and remote user sessions are disconnected after 20 minutes of inactivity:

```
--{ * candidate shared default }--[  ]--
# info with context system aaa
    system {
        aaa {
            authentication {
                idle-timeout 1200
            }
        }
    }
```

5.11 Configuring GRUB password

Procedure

GRUB password protection prevents unauthorized users from modifying the SR Linux boot options present in the GRUB configuration file, `grub.cfg`.

GRUB password protection can be configured in two ways: using a CLI command or by returning the password in BootZ.

Example: Configuring GRUB password using CLI

The grub password is configured using the **system.boot.grub-password.<value>** command. The password itself is provided in clear text. For example, `system.boot.grub-password.srl_test_password`

```
--{ * candidate shared default }--[  ]--
# info system boot grub-password
grub-password grub.pbkdf2.sha512.10000.79250dfc0bd1dddbe0e9588cbd8d21a8da755b70d
7b7ed5246f95847793b87dbdd63fcdd0c9b38cda4fb46e4cdc54ac86df9cd7e3c62a3e6d2db81
661ffff3d0.6545338c7f57e0d62d85b38ecc6b271e32f796ed92d4f367491a85ce83b21203f0
e06f3c953e6edca4713073f94cc4dc7727789f3e3238c93a5869765fa79a93
```

Example: Configuring the GRUB password using BootZ

The clear text GRUB password must first be hashed by the operator using the Linux **grub2-mkpasswd-pbkdf2** command. The resulting hashed password is formatted `grub.pbkdf2.sha512.<iterationcount>.<salt>.<hash-result>` and returned via BootZ.

6 Disk encryption

Disk encryption protects sensitive information in configuration files, logs, and more generally, files located on disk partitions from unauthorized access at rest.

SR Linux uses dm-crypt kernel module block device encryption and LUKS to encrypt the system partitions using AES-XTS-plain64 with a 512-bit key size. The partitions encrypted are `NOKIA-ETC`, `NOKIA-DATA`, and `NOKIA-OPT`, located in the SSD or SD card of the control card.

The disk encryption key is stored in the Trusted Platform Module (TPM) of the control card and is unique per TPM. The key is read from the TPM to decrypt the disk partitions on boot. As each TPM uses a unique encryption key, the encrypted disk cannot be decrypted using a different control card.

Supported platforms

Disk encryption is supported on the following platforms:

- 7220 IXR-H5
- 7250 IXR-6e CPM4 with Root of Trust
- 7250 IXR-10e CPM4 with Root of Trust
- 7250 IXR-X1b
- 7250 IXR-X3b
- 7250 IXR-18e
- 7730 SXR

6.1 Activating disk encryption

Procedure

SR Linux supports activating disk encryption per control card. The activation requires a reboot of the control card. During the reboot, the partitions on the targeted control card are automatically encrypted without data loss.

To activate disk encryption, use the following command:

```
# tools platform trust disk-encryption control <A|B> activate
```

Example

The following example activates disk encryption on slot A:

```
# tools platform trust disk-encryption control A activate
```

The following example shows the warning messages and prompt returned when activating disk encryption:

```
```

```
WARNING: Please proceed to manually reboot the control card for the request to take
 effect. Disk encryption is applied during the control card initialization after reboot.
 This request clears after 5 minutes in the absence of a reboot.
```

Note:

- Disk encryption is not activated by default.
- To deactivate disk encryption, perform a factory reset of the control card.

6.2 Checking disk encryption status

Procedure

Disk encryption status is available per control card and per partition within the control card.

Example

This example displays the disk encryption status for a control card.

```
A:Dut-A# info with context from state platform control A disk-encrypted
    platform {
        control A {
            disk-encrypted true
        }
    }
```

7 Management servers

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

- gRPC server – allows external gRPC 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 gRPC and JSON-RPC management servers.

7.1 gRPC server

Note: gRPC server configuration is supported on the 7730 SXR, 7250 IXR, 7220 IXR, and 7215 IXS.

SR Linux can enable a gRPC server that allows external gRPC clients (gNMI, gNOI, gNSI P4Runtime, and gRIBI) to connect to the device and modify the configuration and collect state information.

When the gRPC server is enabled, the SR Linux gRPC server functions as a target for gRPC clients. The gRPC server validates gRPC clients and passes RPCs to the SR Linux mgmt_server application via the gRPC interface.

Configuration changes made by gRPC 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 gRPC clients and the SR Linux device can be encrypted using TLS. You can specify TLS settings within a TLS profile, and apply the TLS profile when configuring a gRPC server within a network instance. When the gRPC server is enabled, gRPC clients connect and authenticate to the SR Linux device using the settings specified in the TLS profile.

New connections between gRPC clients and the SR Linux device are mutually authenticated. By default, the SR Linux device validates the X.509 certificate of the gRPC 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 gRPC client, performs local authentication if the **metadata-authentication** parameter is set to **true**. In this case, the gRPC client is required to provide a username and password in the metadata of the RPCs. The supplied username and password are authenticated by the SR Linux aaa_mgr application.

In cases where the client's X.509 certificate contains a SPIFFE-ID, the SPIFFE- ID is checked for a match with any user in the system. When a match is found, the client receives the permissions granted to that user. The user is rejected if there is no match. For information about using SPIFFE for client authentication, see Using SPIFFE for client authentication (mTLS).

For more information about supported gRPCs, see the following:

- *SR Linux System Management Guide* (for gNOI, gNMI, and gNSI)
- *SR Linux gRIBI Guide*

- *SR Linux P4RT Guide*

7.1.1 Configuring a gRPC server

Procedure

SR Linux supports configuring one or more gRPC servers for one or more network instances. Each gRPC server is distinguished by a unique configurable name. For each server instance, you can specify the services that the instance supports.

Note:

- If no services are configured when the server instance is enabled, all services are enabled by default.
- For upgrades from pre-24.3.1 releases, gRPC server instance **mgmt** is automatically created and all existing configurations are moved to this path.

You can limit the number of simultaneous active gRPC client sessions, as well as the number of connection attempts per minute by gRPC clients. You can also specify the IP address and port for gRPC client connections, as well as the TLS profile used for authenticating gRPC clients. See TLS profiles.

Example

The following example shows a configuration that enables a gRPC server running gNMI on the SR Linux device. The gRPC server is configured so that gNMI clients can connect to SR Linux via the mgmt network instance on port 50052 (default: 57400). Connecting gNMI clients are authenticated using the settings specified in the TLS profile `tls-profile-1`. The **max-concurrent-streams** parameter sets the limit on the number of concurrent gRPC streams. In addition, the **rate-limit** sets the maximum number of RPC calls per minute while the **session-limit** sets the maximum number of simultaneous active gRPC sessions. Both the **rate-limit** and **session-limit** parameters apply for both synchronous and asynchronous RPCs.

Note: 57400 is the default gRPC server port value, Any other value requires that you configure an ACL CPM filter to accept traffic on the configured port value (in this case 50052).

```
--{ * candidate shared default }--[  ]--
# info with context system grpc-server mgmt
    system {
        grpc-server mgmt {
            admin-state enable
            timeout 7200
            rate-limit 60
            session-limit 20
            max-concurrent-streams 100
            metadata-authentication true
            tls-profile tls-profile-1
            network-instance mgmt
            port 50052
            trace-options [
                request
                response
                common
            ]
            services [
                gnmi
```

```
            ]
            source-address [
                192.168.0.1
            ]
            gnmi {
                commit-confirmed-timeout 10
                commit-save false
                include-defaults-in-config-only-responses false
            }
            unix-socket {
                admin-state enable
            }
        }
    }
```

7.1.2 Configuring maximum paths per subscription request to management servers

About this task

You can define the maximum paths that are supported in any subscription request to the mgmt_server or oc_mgmt_server (for example, for gNMI Subscribe RPCs or CLI **monitor** commands).

Procedure

To configure the maximum number of paths that a single subscription request can subscribe to, use the **max-paths-per-subscription-request** command. The default maximum is 36.

Example: Configure maximum paths per subscription request

```
--{ candidate shared default }--[  ]--
# info with context system configuration max-paths-per-subscription-request
    system {
        configuration {
            max-paths-per-subscription-request 50
        }
    }
```

7.1.3 Disconnecting clients from a gRPC server

Procedure

You can use a **tools** command to manually disconnect gRPC clients from the server.

To do this, obtain the identifier for the P4Runtime client using the **info from state system grpc-server** *<name>* command, then enter the following command to disconnect the client:

Example: Disconnect client from the gRPC server

```
-{ running }--[  ]--
# tools system grpc-server mgmt client 4053 disconnect
```

7.1.4 Displaying gRPC client information

Procedure

To display gRPC service information for a client, use the **info from state system grpc-server mgmt client** <*id*> command, with the following optional parameters.

- **election-id**: election ID of the client
- **gnmi**: container for gNMI related session info
- **gribi**: container for gRIBI related session info
- **p4rt**: container for P4RT related session info
- **remote-host**: remote host of the client
- **remote-port**: remote port of the client
- **rpc**: the called package, service, and RPC
- **start-time**: time of the subscription creation
- **type**: client type
- **user**: authenticated username for the client
- **user-agent**: user agent used for the client

7.1.5 Displaying the RPCs that clients are using

Procedure

To display the RPCs that a client is using, get the identifier for the gRPC client from the **info from state system grpc-server** <*name*> command, then enter the following command to display the RPCs for that client:

Example

```
-{ running }--[  ]--
# info from state system grpc-server mgmt client 4053 rpc
```

7.1.6 gRPC tunnel

gRPC services are based on a client-server model that assumes clients can always initiate and establish connections directly to the gRPC servers. However, in many cases, access control lists (ACLs), network address translation (NAT), firewalls, or other forms of network segmentation can prevent clients from communicating directly with the servers.

To bypass ACLs and NAT devices without modifying their configuration, SR Linux supports the gRPC tunnel service, which allows a gRPC client to communicate with a server over a TCP-over-gRPC tunnel. The gRPC tunnel concept is defined by OpenConfig (see https://github.com/openconfig/grpctunnel) and is based on three entities:

- **target**
  The target represents the network element.

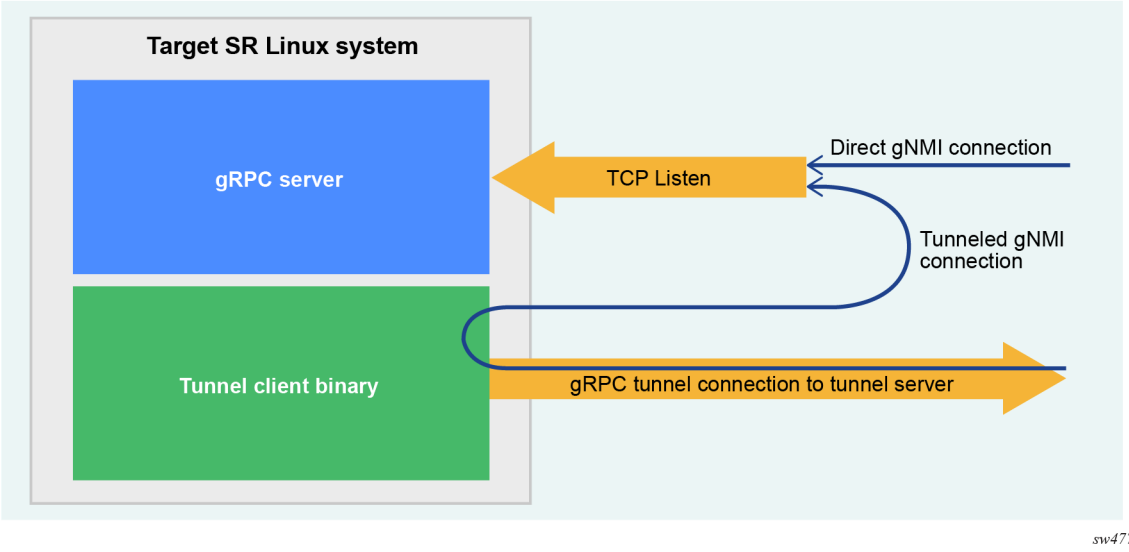
- **tunnel server**
  The tunnel server represents the software entity that tracks registered tunnel clients.
- **tunnel client**
  The tunnel client is a software entity that performs client tasks including registering with the tunnel server.

Target-side gNMI tunnel client as a separate binary

SR Linux implements gRPC tunnels using target-side gNMI tunnel client as a separate binary. In this approach, SR Linux acts as a gRPC tunnel client, where target registrations can only be initiated from the SR Linux tunnel client side. This allows SR Linux to expose its gRPC servers to a gNMI collector through the gRPC tunnel.

In this case, when the tunnel server (gNMI collector) sends gNMI requests to the SR Linux tunnel client, the tunnel client forwards the gNMI requests to the gRPC server's UNIX socket. The gRPC server handles these requests in the same manner as a direct gNMI connection.

Figure 2: Target-side gNMI tunnel client as a separate binary



The same logic can be applied for other target types in SR Linux, such as SSH, NETCONF, gNOI, gRIBI, and P4RT.

For more information, see: https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmignoissh-dialout-grpctunnel.md#target-side-gnmi-tunnel-client-as-a-separate-binary.

Supported RPCs

To establish gRPC tunnels, SR Linux supports the following bidirectional streaming RPCs:

- Register RPC
- Tunnel RPC

7.1.6.1 Register RPC

The Register RPC creates a persistent streaming RPC between the SR Linux tunnel client and the tunnel server.

Register RPC structure

```
rpc Register(stream RegisterOp) returns (stream RegisterOp);
```

The RegisterOp request can be one of three message types:

- **Target**
  In the SR Linux implementation, the target message is sent only from the SR Linux tunnel client to configured gRPC tunnel servers (such as gNMI collectors). The target message registers configured targets that are locally reachable and deregisters targets that are no longer reachable. SR Linux sends a target message to each configured tunnel destination at startup for each configured target, and again when a new target is configured.
- **Session**
  Session messages are only sent from the gRPC tunnel server to the SR Linux tunnel client to request the creation of a session toward an already registered target. The session message is sent after the initial target messages are exchanged. The session is target-specific and, if successful, triggers the establishment of a tunnel from the SR Linux client using the Tunnel RPC to carry a TCP session.
- **Subscription**
  Subscription messages are used to subscribe to notifications about target additions and removals on the other side of the tunnel. However, a gRPC tunnel server (collector) is not required to subscribe to targets for the SR Linux client to send target update events. Instead, the SR Linux client always updates the server about which targets are reachable and which are not.

7.1.6.2 Tunnel RPC

The Tunnel RPC establishes session-specific data tunnels for the actual exchange of data (in the form of TCP datagrams tagged with a tag ID agreed upon during the session registration phase of the Register RPC). The exchange of data over a Tunnel RPC is initiated by the service used by the tunnel. For example, in a gNOI service, the controller opens a gNOI RPC that is tunneled through the gRPC tunnel. The network element handles this request like any other request received directly from the gNOI client.

Tunnel RPC structure

```
rpc Tunnel(stream Data) returns (stream Data);
```

7.1.6.3 gRPC tunnel security

The OpenConfig gRPC tunnel specification requires TLS encryption at both the tunnel and gRPC server level. TLS encryption at tunnel level is configured by assigning the TLS client profile at the destination group level.

When an external gRPC client connects to an SR Linux gRPC server through a tunnel (instead of through direct TCP connection), the source address of the tunnel is used as the IP address to generate certificates.

The TLS server profile assigned to the gRPC server must point to certificates that were generated using this IP address.

7.1.6.4 Configuring gRPC tunnels

About this task

To configure gRPC tunnels, use the system **grpc-tunnel** command.

Example: Configure gRPC tunnels

```
--{ candidate shared default }--[  ]--
# info with-context system grpc-tunnel
    system {
        grpc-tunnel {
            destination destination-name {
                description destination-description
                address 10.1.1.1
                port 50051
                network-instance mgmt
                tls-profile tls-profile-name
            }
            tunnel tunnel-name {
                description tunnel-description
                admin-state enable
                destination destination-name {
                }
                target target-name {
                    id {
                        mac-address
                    }
                    type {
                        gnmi-gnoi-server grpc-server-name
                    }
                }
            }
        }
    }
```

7.2 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.

In cases where the client's X.509 certificate contains a SPIFFE-ID, the SPIFFE- ID is checked for a match with any in the system. When a match is found, the client receives the permissions granted to that user. The user is rejected if there is no match. For information about using SPIFFE for client authentication, see Using SPIFFE for client authentication (mTLS).

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.

7.2.1 Configuring a JSON-RPC server

Procedure

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 TCP port 443. HTTPS requests are authenticated using the settings in the TLS profile `tls-profile-1`.

```
--{ * candidate shared default }--[  ]--
# info with context 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 [
                        ::
                    ]
                }
            }
        }
    }
```

Note:
To connect to the configured JSON-RPC server, enter the <*source-address*> defined in the **system json-rpc-server** configuration, followed by `/jsonrpc`:

`https(s)://<source-address>/jsonrpc`

7.3 TLS profiles

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 gRPC 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.

7.3.1 Configuring a TLS profile

Procedure

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 with context system tls
    system {
        tls {
            profile clab-profile {
                key $aes1$ATMxOVggIuZnwW8=$c7pnl/gCx62+/gDmsIklg8cQW6RbdwxL2Lx
TsrxlMEAtdGupPrjL8wrqvhEWb0ET4Q1hAz4CnwwbcWDdJpOWcV/EzVeRvdR5Rkdt0DemjPXs
Ctbxjxtf/dxmXUOTSLkk80SWNGp97xUMojdhAJUMVDFloU7qpcIYyMrzlsvU3pMCG4VJAF+PPsi
XeUZUz4HnqahtxMcOmCBdLqVgt1iw7kTss/BVqTvpLftXGDyUvALOmgLvvSjPY8fvtC8V1rlpOvWfvXhb
YwrfkYU2pnbkxvLCDDbabgpPD1jUyvBQ4SQz0c3MqXfg8kiWUd0jKw1OM9R6EqcGt6RFuYHON4CRzRmsA/
gUZXbUhUwter8Dz1sJ9D1oU4SkaiJju4RzTIxzBcXtNpar31aBYn2mikCgotV4r+XgcQZsjHqxG2o2wt6e/
VNL+S8bG+1AVG8Tpne/HFmV7wMzWjLP4bWTe0HB3f4pzmN11r0Z87yLGwaCTyXSowy+IYFM1Ffpx8NLWcgH9n
OkvRdDA2xuKFqHYpWIU2UFujRolTe8PifrSeZkSIcwwQcjrs6qbG5qb4xT0gQS4ODGGXpsqeSyRm0FRsU0MRRc
KcGEVeyjDODYNU/PeBnLat598M3NQ8qRLdksN/MSJqehHl4hQCqXtLUiz00PfHy16xCsJX3eUTJ/sv
ByfNsOE//e6PtC5jVW4ATYiZnzIsY8wU0L2xy+hX0TLHz1G9OH5FYqy9lbujBRTFRw5/5Pno7euSB8c
Sub/QLMcbBubDUExSpifrmURP+BREDbsPDL2tFhYgwqal2iYuAQyFE8J3Nmly9iyyZEQZMl/ZqSEm0d5l
PaF6333YkdWW1Dzj85twbPT8Sy3bshhouhAIsnj5Wjk1ToztJZ2CLJdRQ/U8rukbldhdIF+O/6hHNwazheGI
+G2/cE8DpV4kBan80Ty9EZ/dxSoPoELWCTeDBqbr32ADYtF4lRcrgoNcN+jZLnDDBstdz7pKtPb5CO7Vamg
UmptQWDyGiUM/QCuCzpmNlHsT1hcDfORX01xmC0rqq+GX3Qoi4FsbAhYKQKe1wpEE4bvbf5Xdmfb8Y4FMAEDj
Rp98XneIlHIEvHYdlZu6MxB9pL9RMk/hSlsUiyviupvoAAhd0H51MjuRMwDl6K/g5BoZsKW2oZaw/
tcgHPqlopSMIlDV+d8LDax7Zumf2uGxYhRxN/z3IrctkRKUli13exJ9Vybd8NgXXhfyVlg5CR/
Bed9VdiSHTtAWcZxya8wdgeeaEGXgTC+zK9oh9DoT8D3fFAsymedwdmuHCPeWwFWf8xOaxCoZa/
qo3cJrKP2hpqPiFGJRBFxddIi7b5BlWNj4VSC4hof7sdflEEfNQ8jxLeurlGmKr2adLEeODLirgh
Sc2HtZ5i9qtaNLi08Lc30T+xz4SRZNCiX62E3XRkyRyOMMKvU+3u39+oo94PAzsUwwUMvYTQdHglu
ElqU/gEqybmZMqiXp+l6driapo1RlqvEYD7hPTXmNuIwSJOyZspr12eGzEg7DzfBrHNVgWgk2sXExmtW
+mS0nBPX/+k18LQnwgtskD//uamjwC/IKm6zKp8HIKpnTB4qd4nvyKjWiwoh3O9IQqvPrQ7Dn5ihQMM94Nx
ToAqxubIBMCpjcQSLfEOwy0cZVdgSiS/GRfA45UUViAutoSAE/BN+9FQ4UHZa/+3fc1tBtmLm/qs
KwcSn1QyUtaY5V6NVrHW6yiFysAKbnyAgdI5SlXX1oSXQBQvzptoSKRMu9RKGsP/9HaS+qDmuePIj/
UQdlkTU/tOQom2NsleOXBLZFWVUAF+uhq4Ef/ggQNEOaUE8PenAlfhkeaOlNN9XGbDIOEHs/
dQLONLMABgIVWs5E2mj8a2kC3BjyXIdLtGg2a1XDi57ZeT0eH88tra5hKN7+P8O3X8A1llGx/60Q1/
7GsTSYJHN+FQqmgASfvHnZRgcZNPKAutqjuwq4CVngZcoUxOummdM2aW/H5bsl6m8bgJpGPFFK+8REg61v/
cttv7CWc6syempQ16k56iCXmf6yvY6Mloyij9jBRt1fJLqDE67JBNyRr+rstcOpj0fLIsWld8gqtx19dD3h4GJve
TzGlMe3VZuAsnAv8v31SYwwDMFwypO+xacBjO3Pzht++TrVrwLG9R5HPUTnp2lxOQA33Ym3R
+2i65ktKWcnrUMcQInxAm6zwhMA+yQaCZmDIZAfRlZ54wE9mlaATGoI0N7/tw
                certificate "-----BEGIN CERTIFICATE-----
MIID2DCCAsCgAwIBAgICBnowDQYJKoZIhvcNAQELBQAwUzEJMAcGA1UEBhMAMQkw
BwYDVQQHEwAxFTATBgNVBAoTDGNvbnRhaW5lcmxhYjEJMAcGA1UECxMAMRkwFwYD
VQQDExBzcmxjZW9zMDEgbGFiIENBMB4XDTI0MDIxNjA1MDYwNFoXDTI1MDIxNjA1
MDYwNFowVDEJMAcGA1UEBhMAMQkwBwYDVQQHEwAxFTATBgNVBAoTDGNvbnRhaW5l
cmxhYjEJMAcGA1UECxMAMRowGAYDVQQDExFzcmwxLnNybGNlb3MwMS5pbzCCASIw
DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANKb/Um7TzljvC84vpHKt3FwA9/p
```

```
zo94Je1treFJDaH5sVIfHCkGFj0QUcuB9Xc/zEd+O18pY1Xt1/4GdXoYnWlPkXOV
NVJqXt8As/sZOoa4gGTizz0KLOwd/by/FEhMCfIXKOa+5FIC3NgA2LYMYHMTAFYq
EBYku6rcVM96yW4BFnDhatefJf8SFO3wh4AWAQok52H+DxME7rwnBK7B3ITPQKYG
seZZHMC8O65wIHjrUxn+D63Aea6PbJzFlAzT1MaUw3qfMKiErLLXETXg9ylLGugM
TwV9Jh4KwuJTLJch+adhLRIxwipQ+sHtsHG8vjuU8xe+EdgeA4DzMRXNYZsCAwEA
AaOBtDCBsTAOBgNVHQ8BAf8EBAMCB4AwHQYDVR0lBBYwFAYIKwYBBQUHAwIGCCsG
AQUFBwMBMA4GA1UdDgQHBAUBAgMEBjAfBgNVHSMEGDAWgBSBDbZBxouhQI85egna
FbecTUl/2jBPBgNVHREESDBGggRzcmwxghNjbGFiLXNybGNlb3MwMS1zcmwxghFz
cmwxLnNybGNlb3MwMS5pb4cErBQUA4cQIAEBcgAgACAAAAAAAAAAAzANBgkqhkiG
9w0BAQsFAAOCAQEAaCAzkQkjxx32lvWa78pnUzIZmIUr6K1ZciMeslLuQb1PB4jn
GPiKNXghesVoiAmvUV2H7BBf6fLgVQvRqcSgJwyT50MAyRgaHW8L7XnSJVvXpjMU
uLKyvkUWeWZqWzfy6+nA/YhC1qr3FV3/v9s1BbdyBy/eluRcjpQZRrvNKG4+ZAsZ
JR8yWs3k8fTvybpFxcy7V5W7/oVdcw36N17fmA91X5NElk3reEQagmno7GwLOFfx
b8GpxIvu1JjIUDib0Ri8atZlsymwDlCvPPmhFgixlEwVlsAqmhNDBCfmSlmOvROC
eZT2SM6EUlZ4jOvcKFXtDptmMiGeNg9r0Fp3kA==
-----END CERTIFICATE-----"
                authenticate-client false
                cipher-list [
                    ecdhe-ecdsa-aes256-gcm-sha384
                    ecdhe-ecdsa-aes128-gcm-sha256
                    ecdhe-rsa-aes256-gcm-sha384
                    ecdhe-rsa-aes128-gcm-sha256
                ]
            }
        }
    }
```

Note:
The following TLS 1.3 ciphers are always enabled and are non-configurable.

- `tls_aes_256_gcm_sha384`
- `tls_aes_128_gcm_sha256`
- `tls_chacha20_poly1305_sha256`

By default, the following TLS 1.2 ciphers are used unless explicitly configured by a user.

- `ecdhe-ecdsa-aes256-gcm-sha384`
- `ecdhe-ecdsa-aes128-gcm-sha256`
- `ecdhe-rsa-aes256-gcm-sha384`
- `ecdhe-rsa-aes128-gcm-sha256`

7.3.2 TLS certificate tracking and notifications

You can track the validity of certificates configured in TLS profile certificate leaf and receive notifications when certificates are nearing expiration or when certificates have expired. The output of the **info from state with-context system tls profile** command displays the following parameters:

- **expiration** indicates the date and time of the certificate expiry
- **valid-after** indicates the starting date and time of certificate validity
- **expired** indicates whether the certificate is currently expired

When the certificate expiration date is within 30 days of the current system time, the **tlsProfileExpiresSoon** event is triggered. The `Certificate in TLS profile $tls_profile$`

`expires at $expires_at_date_time$` message is generated every 24 hours starting 30 days before expiration and every 60 minutes after the certificate is within 14 days of expiration.

When the certificate expiration date is in the past, the **tlsProfileExpired** event is triggered and the `Certificate in TLS profile $tls_profile$ has expired` message is generated.

7.3.2.1 Viewing TLS profile certificate expiration

Procedure

Execute the **info from state with-context system tls profile** command and specify the name of the profile to view the validity and expiration of the certificates configured in a TLS profile.

Example: Viewing TLS profile certificate expiration

The following example displays the expiration and validity information of the default TLS profile:

```
--{ + candidate shared default }--[  ]--
A:root@srl1# info from state with-context system tls profile __default__
    system {
        tls {
            profile __default__ {
                key $aes1$AW57LbyfJKbgGm8=$gYI8dF7cw/YLdVGUgtX/oVRbi2AyhIs7j
XuJ3vxc1KytsqNttdrzKiTaNTm9ADM8yzohX4/qu6XOef3mINcPutgbAkQ1fC9MXUFuKQ68o38b
+lUI4bfjwr7DQGvpt5oXejLPoiUHJqQGCm0tAYu50th3LOnyLabHACRUfiExMfLhmVXcX4VPuY7htx
Dh8mYTaLSucH2ICiYy4dt0QLoNFsZ5f+gBcH/oQisOmuV6hzb45VG/fjX3fZrcFsIN212j8kd
+HJtFYqvAD1uUWHvHVvOUJix8HKkz7uw9scrq7v021xEEYGR4K9MQmOcigNkhdSn1EPrBbsb
TqZwITxbYpEAPgZrYo+CD9KMQcConmPr9DaGMxGUj0xKBHV+07+nQbj0O8+X9XmvcpReM
+Sn1eP9Shrj4005za5PAPAERWeP4YCgrf0TYksZIqXj8laiVre96m65WIR+6mBHBNRpCrYPp
Ab8uj2flvmG0w5D8F9/ZsQwbIpkHvcQGfk/PEcqnc0h949rdgNPdJ5tRofEBg1z1OIcJOXjMNbDbHRfu282YMVxu
Qq+ASBEs8N1gXzNJIjPaumIOweXEqpdvjkLYGggj3akts9/7PhcAS57TS6RuqRl7jnxSnPVFYgyqNy6sPuBmv3RRJr
CqPxfQYlKfnpfKwB2yaNKNzM65Rlo4j7nhBDFlXsCbDZZLeaEgBbhZwBvM0Z7NlpGahMI8leagZTorY7ActllHGj
QfnFmk/sEWv6TFMlaguOFwob1xDpCzdr3GPeb9NA0irxBDEA7NoEFlxeHL/MSy3XsvSd74fOyWIu
JjlP2LTiG9NI4dCYgiQ4eRc6vXXTMfPcj+csX0QSpSqRPeFINwJ7+4yhNOVAYQiOxXPopfLkD18dl2Wwms
MaAV246nta7OIK9iDLuOpDw+yOU/kBTHYxTf9GtvXeLdEAvdRoQBMEUotVmz80Sqx1D/Y2jytrvcgwbYL5u2f
+cH4EDCSbfQIZNwFh5LcdbYkM9YfxBPGMEBzLywtoMd2KWn/c4hZ5g/pDLrX80xkhX87Zvbvb3NU0igsTV1g
Zf7/K8I1IgzPejeMc72Ehf1oC3FJUpRYbV5GASbYxISkUG9HwcK9jUU5vLgH0lSaav6+JpsJH8jui
AravwUA0OLojwWsb/93zaGE4hCDtCO2ucFbhn17WEqN4tWVBFfoVJlHIZr4apaDNEXchlPFHcgwPMGlb8uR/
oEqnINPs9QAMV8MEbPaujOvIj/IBr9h7Mrl/ZOYTewWHDQC0FBVDPrSNQD27345IrrarBrHfh
+oZPoUSRVBA6A5Eu1tq9hfKSeX+1wkBdEWWn10R/hKYUpz85Al8KpAZInuwxzKfPYSyswkx0xLaH3KH6UkMWDy
Ahva6rOq2IcUq4bK8qW2/x+qwyJSQeCIgs2/D5NqgRrNDBCvL9R4OwXBEGhzTRetGeseYBF7cVnMdUlCI6dD/5NqnI
+x3w81w6PzC6VxrB8JYWq+Lok88am2yRMIcOuftDY6k3wgY4eI4pBWQZ3NLlMPmL5CI7BDDx8725rxN5dCVWxJE8q
YkCznI19pR6Jdcb0ObLOAWL8YSP1Jpx1zEw1m631cR43pZGYoiI478CTvNkpzrrvMAZ5AoVShYARmgco
AkRT9zDBXsdqvF5fQ6B/lNg+OQXrTsZF/bTk4YkvUbR4xuKiuj6N+KznV6dZgnglXB7JdMci30RR5/+776AALcb
WeGS4kvEWVfbhRtNuKujcl2deGN4Du1guI7Q3WJKG9WlGM3GSNvweBLmp0gq3LE9oirclOYBSe5c8lh7u
FiFxHeIdDWgFGwltUAM0sC30w7QlhpzwjIRXnQV1PNc5QY8AZu+Jr8PFEx4EZTtNi83CKAnP/
4RscWHAnv7olY0IhSwC6jIJ5lDdDDF1xWxuoTAQfqYn2xBf6NRNf7SAyLIOEg88HX2cvbIdc8a
Ly1mV3+0bdGDjO5HZJKcls7EBsfzKRMtNI98Mmu2npLqbD9imca7DGiF1UTcq2EoRhO9Xg0d8Jv7+OXjf67z9YC5o
EpExvTv5MnLVX5juPmaH7GkoWGzsrAxmBLidUMoG5bSJ/UXLYaEexFIVNxgT3YDkzFejsvjTv9NOfhGhfdku
TcM7sDpHjAHN9omgvQ7FXvdfjxyEqDYtB6E+UR9H2P1jhl3kKhe1U2nZtR4JfS8mDyjmH9ekAnCLyfWiaS4QD4j
DhYssYKvAoV+33MjOsTZe6KzUodp2nWYLUZvAy4GoTBey8LhVzhohS1r821HBkWokBhkdXRkQSad6R2zKYYf
RyBd+rYVYy7tnG85YClJHqY/6eNBQvry3re/GWXPigk8YgdxPe0ayTsxaiBYGpXZrT5+gWwvtxXv4Os
DsrVoSKSZjtSW2L2YfTbhP24/qfKNLvszbp8jmK8kw3SLnocdyjVXOBn6i+76Y8zKdz0D3pCeDrWahdOAC
+O3qPXEME4f9fzaLv/QvixsbwKoJ3ssRvhFnxwrS0jcumNy+k96mFQZNBZ8Ig4Uh2iUVLrV71VMqz
JrNDpeF/Kx5x08noRT/UQeLhTdbnapQhWckcDJ01uPqv9zri2TFNVboLHtHARqfsv+Awh
LxFQccnO6FWAxp8he042ucoICZlD7UcBxCINM9tcWTrlKCfgIH6gGShZN2WuvDQ4k3zqY42sy8fsB9K6LuEPvEFv
Rqvj7SB/AP36uyRe7B6oLhAqaUluvDMHELBNkNLKiMCs7nKMIrZ/fJsB455om4mwR40cJSJJhVH8Yynt
ZjpPK2qxVWpmFDnTe/75Ns6ELvEtKaHP9PAaE+U1zCNxJeysJFlUazrOiKEgYMNmLASOqO0kVDNidkzsUIz77cpgif
+9PUetQ+ZWM+txCL2KqK1Ztl4L03Gy4AOCOGfFpmWAGfpiXOkbF+36vpZOQGV7o9cE/9C/5tTK0XFtLNOWJh0m
AzhDWkop3giz+xRuX5TDn1jRDmGC75IFrBBDqP87OCd9j8jEvjLcgwZu8BbvdLkT3WHIEcBwbhoS1nklAbQz
```

```
AviuRsdPPqULXLyF6971BtxApAfiQQHgAjTgoCnfjyXXMkXFLnUleDZojWi8GaQdS3YOHxhcJJk1woeSru
+gyLMaQR4+Vq+kY4Av7BYKuxzCkIMLiTv6iNpsgnDSEF8VYPz2oXXCZ8XNKf4CdR0pf1ubjiEq2zZa2TtPh5rqBAw/
sG6Vsj9hGB+eu5WBsYdenYpySOJjmQs2GJKt5vzleq5WmuuyYV0Y676KIOVTjKnP5zjhfq0a9qEN63y61qqp
Oxfb1142RVFEfhVD+JBJRamo7g3gX8/k6FYEQKAe4L5+npvcGb1ZTfz2rWDqKr3ulz/6AWI9PAydJk7aTSTzc8l
MhOCjVgcb0eLgkZ4PR2Jc0ry95gikcVb56xJLNIINQkbo1cyo/swF4EU/AWsrl1YMNPIb7kY3v8Kz3EhXGvK
+H62wftALhxcGS6wAuvBe1l1akV9S2eOWNs63FmMX9SSer7skw9jOkQcfIJNm/bcYif/Xjm2ipQ99LqYLpT5bv
ZuyMjBBg5uobrbz+aJANoJxBtIu4cHlg3RN6AEZnpBpfKxHdBmkwI3Ib9G6rEgCV1zgs9HnScki28Gmy1yoSZZoxl
WrrVrYRj7+R39+jZ3SXuP7Y0zsf9bwnBqXn2UzZN6Wi6of+pIcYqwF0SiguhnQqqioXJpxKdIqO7bAsu
ZifjWJtlqCtlDiY+ka9K4w3pRXKkQS2feudRx1w+dquIGN9ejIXcnlnJ+800EofaMtDfb9Oe7ezuLPNZgS
+sxX7XPgKJhozI+9PzD0nxJL71FXhm5rxWax2vGult8A/2fnOd7nTq+NX6sdgCqw1MgmF1QrdNee
MkbR3hYb+NQ7+4kQOZcJMS2fG1QyzG70kVhRYNbg2Ds+mLjU37NiKnd2s5A9gI+Qtt5pu2unecqq
HuLykyOD9xiqqUiLNf0grtq2kbp43Ni+9nUJUbQbMNnZK+8BjQtAHcg3fXx0sj7QKKSi1IOJ9DKUvdOMww
GxB2VPB6oEOXHRicRODFqjiuzVWLCHjHWkWkVCURPe1wVBaj0M4Ts3ZnTfNbNIE3bm/jeNel/Rcjr1pEIb
JpzIAE2/mg/FQ8rmU419TYbOIWnyoJ7oWgiKUrly3OhSXtMLKw3rjWDTPlGbODA/tbUqg2amEe4pDl96FJ2eCRLQRh
ZbkluDXI5BsC1NIPlMlaZvs+JPcnn+9KhnE5RB3JJRfNPdydzxY9r1spDcbl8DAxableGZ0dmdwF2Uk
LlNr27Vn9JFKqKyc34/8z5TWUpPQxHhsmPG3QnbBtpk33fSGi+t2Lr7HUmt4bIMSU2iZxizT7LU3bMyu9s6YRDg==
                certificate "-----BEGIN CERTIFICATE-----
MIIFLDCCAxSgAwIBAgIUF3nCVJgtq1L+/O/N8dNbunV4UkkwDQYJKoZIhvcNAQEL
BQAwHDENMAsGA1UEAwwEc3JsMTELMAkGA1UEBhMCVVMwIBcNMjUwOTE1MTQwMjI2
WhgPMjIwNTAyMTkxNDAyMjZaMBwxDTALBgNVBAMMBHNybDExCzAJBgNVBAYTAlVT
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAqM84lxKW2bM66Hd9Brxi
i7fkPuwOm39Y0nx73r9T02BKOHZDmOSapRb1LbPcJcKE9u44CUtn9xzuPkhy85D6
2i/J9+XJd03nylXRLy/VvbGxDUkiJ5Av1TR10ali1xmvDf42D8WssG9gH2VoG7JA
tBYo9xjwvHEqPxqp8hTUDbmx4gSfJtDDbkYFWzDPgu405Ad6+Pp1LFwUod0EpBG/
WyQCuwVYGOGwcSUDrt/rIQR37Cdw3VT9hcyIhRMUUgHH5/Khpd39kvb5Uvkmqhw1
M/kynKod1cCetjkPoFYy3ZRoVMCrdO0yWh9/6l680G9o1hECpg6DIPU9Gbp/ILVO
K1hJKR4Ykt66djh7y3k6IPu+XWJ9GQKx/JXZgl86B2eOVaul/TgaGkk5lSIdmIw7
+7nMHavYtZ0Km2fVE9qJiOmrhM71rRyvdVwf536XT8fGw4S1GxvE9DCfIUbamUq0
pcpcBT6oMJP55cxAH/3q6+tRpPXjGvRZ5YWoHGMw3lW4uQNYOOXUaDoU0jw2dfGW
hXamz4rE1lrA5HxKX16cQauZoYmJ6AutdsrVRAqy5Hbgj6J68qgry/267XMc8F8f
NdP3r4BajQoXMR+eO6+ktUy2nmz4785S65qFIuc2M0R8JPhfknfPcYOqiQN79G6G
4uk+uhq1EvPkiwW1K2FCabcCAwEAAaNkMGIwHQYDVR0OBBYEFNi4JsX2IqxZz+nM
DpyABIIx/mmUMB8GA1UdIwQYMBaAFNi4JsX2IqxZz+nMDpyABIIx/mmUMA8GA1Ud
EwEB/wQFMAMBAf8wDwYDVR0RBAgwBoIEc3JsMTANBgkqhkiG9w0BAQsFAAOCAgEA
HqO6wbu94iBURsC3/ki8c/ULwpYe8Z4wIB6hiy3xpsS8UMolF0DF6H6CNBhvC0V5
/VFBmX4qbv+MjTxGuskVYgABm7FzSGibvpE4C/W/xxW7NG4oPRU8eBrnThxL12GF
ssDe/E/JlqGSNYYq95+u39wAvtLtvZz2FJgaF9T/R8yvmj6553PJE6PoBZ81UgDM
onMLzZuBAbHB3qdzKE1CiKcvGQHbRW4UREAzWtC0Kvg1H8BLwycozTX+wNmIO3kd
aFF6ipziUeTYFEEjafkfILBSDixDiP9JTepGdmAGLkJahHbIUD8H+Q4+6gGyMNai
KY98xTul8ZJA2E50iB5aG+wapytcn1RocICIJbNWHn+g20o9qckS+gnJhO3akdOu
VeSEOABhItn+tthxSMuVfrmWJHe3nqvOT9RgMXof14xos4W42UMOnTzksVIHSlu2
B15VQaEYnCzOIPPpqm5z0vAQOkt0VL3aarcs2jIU3qU117+hW2U3F5gaTFFXdjSV
uWu+04LZre7gPCmJuM74nvn/xHuEDf9Ob55HU/kKWlObPpyJcVWBZVP4uhwtAi/h
ps/dWxeeJOiNDLFBkKXxYFrNNtnokkQEC9r58stoB5TUP3sL7gaOh6Gusu/ZIwE4
TTbaeVcGrf9AknKhxLvmkXSzaKKZjX8Doi5Y2x3GsRE=
-----END CERTIFICATE-----
"
                authenticate-client false
                dynamic true
                valid-after 2025-09-15T14:02:26.000Z
                expiration 2205-02-19T14:02:26.000Z
                expired false
                cipher-list [
                    ecdhe-ecdsa-aes256-gcm-sha384
                    ecdhe-ecdsa-aes128-gcm-sha256
                    ecdhe-rsa-aes256-gcm-sha384
                    ecdhe-rsa-aes128-gcm-sha256
                ]
            }
        }
    }
```

7.3.3 Configuring a TLS profile with TPM Device Identity

Procedure

Note: TLS client authentication using the TPM IDevID key and certificate is supported on the 7250 IXR-6e, 7250 IXR-10e, and 7250 IXR-18e CPM4 with Root of Trust , 7250 IXR-X1b, and 7250 IXR-X3b.

SR Linux supports configuring a TLS profile to use the TPM IDevID (Initial Device Identity) or oIDevID (owner Issued Initial Device Identity) key and certificate, instead of configuring the private key and certificate.

Use the **system tls profile use-tpm-devid** command to configure the TLS server to authenticate clients based on the TPM device IDs and certificates during TLS handshake and session establishment.

Example: Configure TLS profile to use IDevID key with IDevID certificate

In the following example, TLS connections are authenticated using the TPM IDevID key and certificate. This profile affects any new TLS sessions that use it.

```
--{ * candidate shared default }--[  ]--
info with context system tls profile clab-profile
    system {
        tls {
            profile clab-profile {
                use-tpm-devid idevid
            }
        }
    }
```

Example: Configure TLS profile to use IDevID key with oIDevID certificate

In the following example, TLS connections are authenticated using the TPM IDevID key and the operator's device ID certificate (oIDevID). This profile affects any new TLS sessions that use it.

```
info with context system tls profile clab-profile
    system {
        tls {
            profile clab-profile {
                use-tpm-devid oidevid

            }
        }
    }
```

Note: When the `use-tpm-devid` parameter is set to oIDevID, and oIDevID is not present, the system does not revert to IDevID. This ensures that the TLS server is operational only if oIDevID is present on the CPM.

7.3.4 Generating a self-signed certificate

Procedure

From the SR Linux CLI, you can create a self-signed certificate and key. By default, SHA-256 hash functions 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
```

7.3.5 Generating a certificate signing request

Procedure

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
```

7.3.6 SPIFFE

SPIFFE (Secure Production Identity Framework for Everyone) is a framework that enables services to bootstrap and obtain identity. SPIFFE enables a central authority (CA) to issue signed certificates for identifying clients and servers, establishing mutual trust among them based on their trust in the CA.

SPIFFE is applicable only for TLS sessions and is not used when system is accessed over SSH/CLI or any other non-TLS interfaces.

SPIFFE defines an identity document known as the SPIFFE Verifiable Identity Document (SVID). This is a cryptographically-verifiable document that contains a SPIFFE ID, which represents the identity of a service. SVIDs can be encoded in two formats: X.509 certificates or JWT tokens1.

Note: SR Linux supports X.509 certificates as SVIDs only.

In an X.509 SVID, the corresponding SPIFFE ID is set as a URI type in the Subject Alternative Name (SAN) extension (`subjectAltName`) field. The SVID only supports one URI field in the SAN extension. You can add multiple IP or DNS fields to the SAN, but there must be only one URI in the SAN.

A SPIFFE-ID is a URI that starts with `spiffe` scheme, followed by a FQDN and a path that identifies the user. For example,

```
spiffe://nokia.com/sa/alice
```

where, `nokia.com` refers to the domain name, and `alice` refers to the user.

SPIFFE is used for authentication only. Authorization is performed by the role assigned to the user referred by the SPIFFE-ID.

7.3.6.1 Leveraging SPIFFE

The high-level steps for leveraging SPIFFE are as follows:

1. Create two sets of certificates, one for the server and one for the client. These certificates should be signed by a Certificate Authority (CA).
2. Configure a TLS profile on the system with `authenticate-client` parameter set to `true` and the `trust-anchor` parameter set to the public certificate of the CA that was used to sign the client and server certificates. For more information, see Configuring a TLS profile.
3. Reference the TLS profile to use on a TLS supporting server. For configuration examples, see the sections that correspond to the TLS server you are using:
   - For gRPC server configurations, see Configuring a gRPC server.
   - For JSON-RPC server configurations, see JSON-RPC server.
   - For gRIBI server configurations, see the section "Enabling the gRIBI server for a network-instance" in *SR Linux gRIBI Guide*.

- For P4Runtime server configurations, see the section "Configuring the P4Runtime server for a network-instance" in *SR Linux P4RT Guide*.

4. Configure a SPIFFE-ID for a user. For instructions, see Configuring SPIFFE-ID for users.

Note: The TLS supporting server does not check the SPIFFE-ID in the client's X.509 certificate unless you enable mTLS (set **system tls profile authenticate-client** to **true**). Additionally, the presence of a SPIFFE-ID in the client's X.509 certificate does not necessarily indicate support for mTLS.

7.3.6.2 Configuring SPIFFE-ID for users

Procedure

The SPIFFE-ID list is configured under **system aaa authentication user**. The format of SPIFFE-ID is `spiffe://<domain-name>/<user>`.

Example: Configure SPIFFE-ID for a local user

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authentication user srl-test-user spiffe-ids
    system {
        aaa {
            authentication {
                user srl-test-user {
                    spiffe-ids [
                        spiffe://nokia.com/srl-test-user
                    ]
                }
            }
        }
    }
```

Example: Configure SPIFFE-ID for a admin user

```
--{ * candidate shared default }--[  ]--
# info with context system aaa authentication admin-user spiffe-ids
    system {
        aaa {
            authentication {
                user admin-user {
                    spiffe-ids [
                        spiffe://nokia.com/admin-user
                    ]
                }
            }
        }
    }
```

7.3.6.3 Using SPIFFE for client authentication (mTLS)

In a handshake with TLS client authentication (`authenticate-client` enabled), the server expects the client to present its X.509 certificate.

In cases where the client's X.509 certificate contains a SPIFFE-ID, the SPIFFE-ID is checked for a match with any SPIFFE-ID configured within the `spiffe-ids` field in the user configuration. When a match is found, the client receives the permissions granted to that user. The user is rejected if there is no match.

If the X.509 certificate provided by the client does not contain a SPIFFE-ID, the behavior depends on whether the TLS supporting server has enabled the `use-authentication` or not. When enabled, the HTTP metadata is checked for the username and password. To proceed with authentication, both the username and password must be provided. When the `use-authentication` is not enabled, the client is rejected.

8 Logging

SR Linux implements logging via the standard Linux rsyslogd package. With dynamic configuration, log filters can be defined and passed to remote servers or other specified destinations.

SR Linux `log_mgr` application supports the standard RFC 5424 message format when relaying syslog messages to remote servers.

RFC 5424 introduces a more structured and extensible syslog message format designed to overcome the limitations of the earlier syslog formats, such as RFC 3164. RFC 5424 includes features like high-precision time-stamping with timezone information and support for embedding structured log data.

The main configuration file for rsyslogd is `/etc/rsyslog.conf`. SR Linux installs a minimal version of the `/etc/rsyslog.conf` file and maintains an SR Linux-specific configuration under the `/etc/rsyslog.d/` directory.

You can configure SR Linux logging using the CLI or a northbound API. Although you can add `*.conf` files manually in the `/etc/rsyslog.d` directory, it is not recommended.

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

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.

Logging configuration consists of specifying a source for input log messages, possibly filtering messages by some pattern, and specifying an output destination.

Templates are easy-to-reuse message formats. rsyslog contains pre-defined templates identified by the `RSYSLOG_` prefix. For information about the supported rsyslog templates, see sections Rsyslog templates for local buffer, file, or console output and Rsyslog templates for forwarding messages to remote servers.

8.1 Input sources for log messages

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

Table 10: Linux syslog facilities describes the Linux syslog facilities that can be used as input sources for log messages.

Table 10: Linux syslog facilities

Facility	Description
all	All supported Linux syslog facilities
audit	Audit messages
auth	Security/authorization messages that do not contain secret information

Facility	Description
authpriv	Security/authorization messages that may contain secret information
console	Alert messages
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
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
ntp	Messages generated by NTP 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 11: 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 11: SR Linux logging subsystem names

Subsystem	Description
aaa	Messages generated by the aaa_mgr application (not including accounting messages)

Subsystem	Description
accounting	Accounting messages generated by the aaa_mgr application
acl	Messages generated through an ACL log action
app	Messages generated by the app_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
fib	Messages generated by the fib_mgr application
gnmi	Messages generated by the grpc_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

8.2 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 action 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 grpc_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.

8.3 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 include or exclude specific prefix, text within a message, or regular expressions. Negative matching uses the `if not` condition from rsyslog, which allows you to exclude specific log messages.

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 performed 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 it is local6), and you can create filters for subsystem-specific messages from this facility.

Table 12: Logging priorities describes the logging priorities in order of severity.

Table 12: Logging priorities

Code	Priority name	Description
0	emergency	System is unusable
1	alert	Action must be taken immediately
2	critical	Critical conditions
3	error	Error conditions
4	warning	Warning conditions
5	notice	Normal but significant conditions
6	informational	Informational messages
7	debug	Debug-level messages

8.3.1 Defining filters

Procedure

Filters target specific messages or groups of log messages in the input message source. You can define the following filter criteria for log messages:

- specific text in a message
- prefix text at the beginning of a message
- Linux facility that generated the message
- regular expression matching text in the message
- syslog tag of the message
- remove specific messages from being logged using the following filter criteria that have negative matches:
  - does not contain specific text in a message
  - does not contain specific prefix text at the beginning of a message
  - does not contain specific regular expression matching text in the message

Example: Match kernel messages with warning or higher priority

The following example shows a configuration that creates a filter that matches messages from the Linux facility kernel that have a priority of warning or higher. See Table 12: Logging priorities for a list of logging priorities.

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

Example: Match informational or higher accounting messages

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 with-context system logging filter f2
    system {
        logging {
            filter f2 {
                contains accounting
                facility local6 {
                    priority {
```

```
                        match-above informational
                    }
                }
            }
        }
    }
```

Example: Exclude informational or higher accounting messages

The following example creates a filter that excludes 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`.

```
--{ + candidate shared default }--[  ]--
# info with-context system logging filter f3
    system {
        logging {
            filter f3 {
                not-contains accounting
                facility local6 {
                    priority {
                        match-above informational
                    }
                }
            }
        }
    }
```

8.4 Logging destination configuration

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

- log file
- memory buffer storage
- console (`/dev/console`)
- one or more remote servers

8.4.1 Specifying a log file destination

Procedure

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 four files).

Example: Send critical cron messages to a file

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 with context system logging
```

```
    system {
        logging {
            file cron-critical {
                facility cron {
                    priority {
                        match-exact [
                            critical
                        ]
                    }
                }
            }
        }
    }
```

Example: Send messages matching a filter to a file

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

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

Example: Send specific message types to a file

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 with context system logging
    system {
        logging {
            file aaa-warn-info {
                subsystem aaa {
                    priority {
                        match-exact [
                            warning
                            informational
                        ]
                    }
                }
            }
        }
    }
```

8.4.2 Specifying a buffer destination

Procedure

You can configure SR Linux to 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 with context system logging
    system {
        logging {
            buffer f1-match {
                rotate 5
                size 5000000
                filter [
                    f1
                ]
            }
        }
    }
```

8.4.3 Specifying the console as destination

Procedure

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 with context system logging
    system {
        logging {
            console {
                subsystem accounting {
                    priority {
                        match-above informational
```

```
                    }
                }
            }
        }
    }
```

8.4.4 Specifying a remote server destination

Procedure

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 with context system logging
    system {
        logging {
            network-instance mgmt
            remote-server 192.168.0.3 {
                   subsystem bgp {
                       priority {
                           match-above alert
                       }
                   }
               }
            }
        }
   }
```

8.5 Specifying a Linux syslog facility for SR Linux subsystem messages

Procedure

All of the messages generated by SR Linux subsystems (see Table 11: SR Linux logging subsystem names) are logged to the same Linux syslog facility. This behavior 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 10: 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 with context system logging
```

```
    system {
        logging {
            subsystem-facility local7
        }
    }
```

8.6 Specifying FQDN for logging hostnames

Procedure

SR Linux allows you to configure the logging system to use either the system hostname or the system FQDN in the hostname field of the message

Example

The following example configures SR Linux to use the system FQDN for logging messages.

```
--{ * candidate shared default }--[  ]--
# info with context system logging
    system {
        logging {
            use-fqdn true
        }
    }
```

When the `use-fqdn` option is set to `true`, the system FQDN is used in the syslog server hostname field. The default is `false`. This ensures backward compatibility with previous releases that only supported using the system hostname for logging messages. The domain name must be configured for the FQDN to take effect. See Configuring a domain name.

8.7 Rsyslog templates for local buffer, file, or console output

Procedure

SR Linux `log_mgr` application supports the following rsyslog templates for local buffer, file, or console output.

- `RSYSLOG_DebugFormat`
- `RSYSLOG_TraditionalFileFormat`
- `RSYSLOG_FileFormat`

Example: Using `RSYSLOG_DebugFormat`

The `RSYSLOG_DebugFormat` template is mainly meant for debugging and diagnostic purposes and must not be used for remote forwarding or production environment.

```
--{ * candidate shared default }--[  ]--
# info with context system logging file messages
    system {
        logging {
            file messages {
                format RSYSLOG_DebugFormat
```

```
                rotate 3
                size 10000000
                facility local6 {
                    priority {
                        match-above warning
                    }
                }
            }
        }
    }
```

Example: Using RSYSLOG_TraditionalFileFormat

The message formatRSYSLOG_TraditionalFileFormat has low-precision timestamps.

```
--{ * candidate shared default }--[  ]--
# info with context system logging file messages
    system {
        logging {
            file messages {
                format RSYSLOG_TraditionalFileFormat
                rotate 3
                size 10000000
                facility local6 {
                    priority {
                        match-above warning
                    }
                }
            }
        }
    }
```

Example: Using RSYSLOG_FileFormat

The default message format RSYSLOG_FileFormat supports high-precision timestamps and timezone information, as described in RFC 3339.

```
--{ * candidate shared default }--[  ]--
# info with context system logging file messages
    system {
        logging {
            file messages {
                format RSYSLOG_FileFormat
                rotate 3
                size 10000000
                facility local6 {
                    priority {
                        match-above warning
                    }
                }
            }
        }
    }
```

8.8 Rsyslog templates for forwarding messages to remote servers

Procedure

SR Linux `log_mgr` application supports the following rsyslog templates for forwarding messages to remote servers.

- `RSYSLOG_ForwardFormat`
- `RSYSLOG_SyslogProtocol23Format`
- `RSYSLOG_TraditionalForwardFormat`

Example: Using `RSYSLOG_ForwardFormat`

The message format `RSYSLOG_ForwardFormat` forwards logs with high-precision timestamps.

```
--{ * candidate shared default }--[  ]--
# info with context system logging remote-server 192.0.2.1 format
    system {
        logging {
            remote-server 192.0.2.1 {
                format RSYSLOG_ForwardFormat
            }
        }
    }
```

Example: Using `RSYSLOG_SyslogProtocol23Format`

The message format `RSYSLOG_SyslogProtocol23Format` guarantees reliable and efficient logging of important system events. This default message format is defined in IETF internet-draft ietf-syslog-protocol-23 and closely adheres to the syslog standard RFC5424.

```
--{ * candidate shared default }--[  ]--
# info with context system logging remote-server 192.0.2.1 format
    system {
        logging {
            remote-server 192.0.2.1 {
                format RSYSLOG_SyslogProtocol23Format
            }
        }
    }
```

Example: Using `RSYSLOG_TraditionalForwardFormat`

The message format `RSYSLOG_TraditionalForwardFormat` forwards logs with low-precision timestamps.

```
--{ * candidate shared default }--[  ]--
# info with context system logging remote-server 192.0.2.1 format
    system {
        logging {
            remote-server 192.0.2.1 {
                format RSYSLOG_TraditionalForwardFormat
            }
        }
    }
```

8.9 TLS encryption support

SR Linux supports encrypting rSyslog traffic using TLS. rSyslog includes support for the OpenSSL driver, which enables TLS based transport. TLS is optional and is enabled only when selected as the transport type by the operator.

To enable TLS, you must create a TLS profile and ensure it is accessible to rSyslog. This profile allows rSyslog to link to the required certificate and key files specified in the configuration.

By default, rSyslog uses port 514 for non-TLS traffic and port 6514 for TLS traffic, unless the user explicitly configures a different port. For a configuration example, see Configuring TLS on rSyslog log events.

8.9.1 Configuring TLS on rSyslog log events

Procedure

To configure TLS encryption, set the **transport** parameter to `tls` in the `system logging remote-server{.host}` context and configure a TLS profile for the remote server.

Example

The following example shows a configuration that enables TLS encryption using the `clab-profile` TLS profile on port 6514. The logs are formatted according to the `RSYSLOG_Syslog Protocol23Format` standard. The **transport** parameter is set to `tls`, specifying that TLS encryption should be used for the connection.

```
--{ * candidate shared default }--[  ]--
# info with-context system logging remote-server 192.168.0.3
    system {
        logging {
            remote-server 192.168.0.3 {
                transport tls
                tls-profile clab-profile
                remote-port 6514
                source-address 192.168.0.3
                format RSYSLOG_SyslogProtocol23Format
            }
        }
    }
```

9 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 typically 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 The 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 the *SR Linux Interfaces Guide* for more information.

9.1 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 with-context network-instance black
    network-instance black {
        type ip-vrf
        admin-state enable
        description "Sample network instance"
        router-id 192.168.2.1
        ip-forwarding {
            receive-ipv4-check true
        }
        interface ethernet-1/1.1 {
        }
        interface ethernet-1/2.1 {
        }
        interface lo0.1 {
        }
        protocols {
            linux {
                export-routes true
```

```
                export-neighbors 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.

9.2 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 system-level **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 with-context network-instance default
    network-instance default {
        mtu {
            path-mtu-discovery true
        }
    }
```

```
--{ * candidate shared default }--[ ]--
# info with-context system mtu
    system {
        mtu {
            min-path-mtu 552
        }
    }
```

9.3 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.

To configure static routes, you specify route prefixes to point to next-hop-groups. In the current release, SR Linux supports two models for configuring static routes to point to next-hop-groups:

- Legacy model – This model uses next-hop-groups to hold the configuration details for next-hops used by a static route. Direct configuration of next-hops is not done in the static route object itself. The Legacy model is supported in the current release, as well as all previous SR Linux releases. The Legacy model is configured under the `network-instance.static-routes` context.

- New model – This model, based on OpenConfig, allows a static route to reference a statically configured next-hop-group, which in turn references one or more static next-hop members. Next-hops are configured as separately indexed objects. The New model also allows a statically configured next-hop to specify attributes for tunnel encapsulation headers that are applied to a packet when it is forwarded to this next-hop.
  The New model is configured under the `network-instance.static` model.

9.3.1 Legacy model for configuring static routes

In the Legacy model, 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). SR Linux supports the following types of next-hops in a next-hop group:

- Regular IP-numbered next-hops

  Matching IP packets are forwarded, without encapsulation, toward the next-hop, with one or more resolution steps deciding the outbound interface and MAC address to use for forwarding. 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.

- MPLS next-hops (7250 IXR-6, 7250 IXR-10, 7250 IXR-6e, 7250 IXR-10e, and 7250 IXR-18e only)

  Matching IP packets are forwarded, with an added MPLS label stack, toward the next-hop, with one resolution step deciding the outbound interface and MAC address to use for forwarding. The resolve flag must be set to **false**, because only a direct route can be used to resolve the IPv4 or IPv6 next-hop address.

- GRE tunnel next-hops (7250 IXR-6, 7250 IXR-10, 7250 IXR-6e, 7250 IXR-10e, and 7250 IXR-18e only)

  Matching IP packets are forwarded, with added IP/GRE encapsulation, toward a remote GRE tunnel endpoint. A GRE tunnel next-hop can be configured with an IPv4 or IPv6 address of the remote GRE tunnel endpoint; this is the destination address in the outer IP header of the IP/GRE packet.

  A next-hop group can have up to 128 GRE tunnel next-hops; more than one next-hop implies ECMP. An ECMP hash calculation using header fields of the payload packet (before IP/GRE encapsulation) selects one of these next-hops and in so doing, selects an outer IP destination for the GRE encapsulation.

  A GRE tunnel next-hop is resolved if the destination address is reachable (using any route type) in the route table. If a GRE tunnel endpoint is unresolved, it is removed from the next-hop group ECMP set. If a next-hop group has no resolved GRE tunnel next-hops, it is not usable and all referencing static routes are removed from the route table and FIB.

  A GRE tunnel next-hop (more precisely, its destination IP) can be resolved by a normal ECMP route or a weighted ECMP (wECMP) route. The mapping of a GRE tunnel next-hop to an ECMP next-hop of the resolving route is based on a hash of the configured destination IP address. Load balancing of the GRE tunnels follows the wECMP programming of the resolving route. See the *SR Linux Routing Protocols Guide* for more information about wECMP in SR Linux.

- Unnumbered IPv4 next-hops

  IP packets hashed to the next-hop are forwarded, without encapsulation, to a specified interface, assumed to be an unnumbered IPv4 interface. The unnumbered IPv4 interface is specified within an `interface-ref` container and must refer to a subinterface associated with the network-instance.

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 Bidirectional Forwarding Detection (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.

Backup next-hop groups

It is common practice for one static route to serve as a backup for another static route for an IP prefix. The backup static route is known as a floating static route because it is not installed or activated unless the primary static route becomes unusable.

In SR Linux, you can configure a floating static route by attaching a statically configured backup next-hop group to the statically configured primary next-hop group of the static route entry. As long as the primary next-hop group has one or more viable and usable next-hops, forwarding is based on those next-hops and the backup next-hop group is not used. If all next-hops of the primary next-hop group become unusable, then the system exchanges the primary next-hop group for the backup next-hop group, and forwarding continues according to the next-hops of the backup next-hop group.

The backup next-hop group is replaced by the primary next-hop group immediately after the primary next-hop group has usable next-hops again. A next-hop is usable if it can resolved to an outgoing interface that is up, and (if applicable) the BFD session associated with the next-hop is up.

The following considerations apply to backup next-hop groups:

- The backup next-hop group can specify a blackhole action. This action causes the static route to remain programmed when it has no viable next-hops in the primary next-hop group; normally, when the primary next-hop group does not have a backup next-hop group, the static route is removed from the route table/FIB.
- The backup next-hop group can have multiple (ECMP) next-hops.
- The backup next-hop group is not limited to having the same types of next-hops as the primary. For example, the primary next-hop group can have GRE tunnel next-hops, and the backup can have regular IP-numbered next-hops.
- A backup next-hop group cannot have its own backup next-hop group.

9.3.1.1 Configuring static routes (Legacy model)

Procedure

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

Example: Configure IPv4 and IPv6 static routes

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 with-context 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: Configure IP-numbered next-hop groups

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

```
--{ * candidate shared default }--[ network-instance black ]--
# info with-context next-hop-groups
    next-hop-groups {
        group static-ipv4-grp {
            admin-state enable
            nexthop 1 {
                ip-address 192.0.2.22
            }
            nexthop 2 {
                ip-address 192.0.2.45
                resolve true
            }
        }
        group static-ipv6-grp {
            admin-state enable
            blackhole {
            }
        }
    }
}
```

In this 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.

Example: Configure a MPLS next-hop group

The following example specifies a MPLS next-hop for `nhop_group_1`. Only one MPLS next-hop is supported per next-hop group. In this example, the label for outgoing traffic to MPLS next-hop 192.35.1.5 is swapped to 1001.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default next-hop-groups group nhop_group_1
```

```
    network-instance default {
        next-hop-groups {
            group nhop_group_1 {
                nexthop 0 {
                    ip-address 192.35.1.5
                    resolve false
                    pushed-mpls-label-stack [
                        1001
                    ]
                }
            }
        }
    }
```

Example: Configure a GRE tunnel next-hop group

The following example specifies a GRE tunnel next-hop group. Packets are forwarded, with added IP/GRE encapsulation, toward a remote GRE tunnel endpoint specified by the `destination-ip`.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default next-hop-groups group NHG1
    network-instance default {
        next-hop-groups {
            group NHG1 {
                nexthop 1 {
                    encapsulate-header gre
                    gre {
                        destination-ip 20.20.20.1
                    }
                }
                nexthop 2 {
                    encapsulate-header gre
                    gre {
                        destination-ip 20.20.20.2
                    }
                }
                nexthop 3 {
                    encapsulate-header gre
                    gre {
                        destination-ip 20.20.20.3
                    }
                }
            }
        }
    }
```

You can optionally specify the `source-ip` for a next-hop in a GRE next-hop group. If the `source-ip` is not specified, the lowest-numbered loopback address of the associated network-instance is used.

Example: Configure an unnumbered IPv4 next-hop group

The following example configures an unnumbered IPv4 next-hop group. The `interface-ref` container must specify both an interface and subinterface. The subinterface must be associated with the network-instance. The `resolve` parameter for the unnumbered next-hop must be set to `false`.

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default next-hop-groups group unhg
    network-instance default {
        next-hop-groups {
            group unhg {
```

```
                admin-state enable
                nexthop 1 {
                    resolve false
                    interface-ref {
                        interface ethernet-1/1
                        subinterface 1
                    }
                }
            }
        }
    }
```

Example: Configure a link-local address in an IPv6 next-hop group

IPv6 next-hop addresses in a static next-hop group can be global unicast IPv6 addresses or IPv6 link-local addresses (LLAs). If you configure an IPv6 LLA, you specify it with a zone, as a string in the following format:

<link-local-ipv6-address>%<subinterface-name>

where <subinterface-name> has the format <interface-name>.<subinterface-index> and corresponds to a configured `interface.subinterface` object in the `interface` configuration hierarchy on the SR Linux device.

The following example configures an IPv6 LLA for a subinterface and specifies it as a next-hop in a next-hop group.

```
--{ * candidate shared default }--[  ]--
# info with-context interface ethernet-1/10 subinterface 1 ipv6
    interface ethernet-1/10 {
        subinterface 1 {
            ipv6 {
                admin-state enable
                address fe80::24:1/64 {
                    type link-local-unicast
                }
            }
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default next-hop-groups group ecmp nexthop 1
    network-instance default {
        next-hop-groups {
            group g1 {
                nexthop 1 {
                    ip-address fe80::25:1:2%ethernet-1/10.1
                    resolve false
                    failure-detection {
                        enable-bfd {
                            local-address fe80::25:1:1
                        }
                    }
                }
            }
        }
    }
```

Example: Configure a backup next-hop group

The following example configures a backup and a primary next-hop group for a prefix. If all of the next-hops of the primary next-hop group become unusable, then forwarding is based on the next-hops in the backup next-hop group. When the primary next-hop-group has usable next-hops again, forwarding is based on the next-hops in the primary next-hop group.

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance black next-hop-groups group backup-NHG
    network-instance black {
        next-hop-groups {
            group backup-NHG {
                nexthop 1 {
                    ip-address 192.0.2.22
                }
                nexthop 2 {
                    ip-address 192.0.2.45
                }
            }
        }
    }
```

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance black next-hop-groups group primary-NHG
    network-instance black {
        next-hop-groups {
            group primary-NHG {
                backup-next-hop-group backup-NHG
                nexthop 1 {
                    ip-address 192.35.1.5
                }
                nexthop 2 {
                    ip-address 192.35.1.10
                }
            }
        }
    }
```

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance black static-routes
    network-instance black {
        static-routes {
            route 192.168.18.0/24 {
                admin-state enable
                metric 1
                preference 5
                next-hop-group primary-NHG
            }
        }
    }
```

9.3.2 New model for configuring static routes

Unlike the Legacy model, where the configuration details for next-hops used by a static route were held in the next-hop-group configuration, the New model places the configuration of the next-hops directly in the next-hop object itself. In the New model, a static route can reference a statically configured next-hop-group, which in turn references one or more static next-hop members.

In the New model, a static next-hop is configured under the `network-instance.static.next-hops.next-hop` context and can be referenced by multiple different next-hop-groups.

A static next-hop configured under this context has all the configurable attributes as a next-hop configured under the Legacy model, as well as configurable attributes related to tunnel encapsulation. The New model allows a static next-hop to add multiple encapsulation headers to a packet when it is forwarded to this next-hop. For each encapsulation header type, there is a container with configurable attributes specific to that type. For example, for the Generic UDP Encapsulation (GUE) type, the following attributes can be configured:

- source IP address
- destination IP address
- destination UDP port
- IPv4 TTL

GUE tunnels can be encapsulated using a static route and can be decapsulated using policy-based forwarding. See Configuring static routes (New model) for an example of configuring encapsulation settings for a GUE tunnel. See the *SR Linux ACL and Traffic Steering Guide* for information about GUE tunnel decapsulation.

In the Legacy model, a static next-hop-group can be administratively enabled or disabled independently from the static route. In the New model, the only administrative control is at the prefix level.

9.3.2.1 Configuring static routes (New model)

Procedure

To configure static routes using the New model, you configure the next-hops, add the next-hops to a next-hop-group, and specify route prefixes to point to next-hop groups, along with the metric and preference.

You can also configure a next hop with tunnel encapsulation settings and reference the next-hop in a next-hop-group. The tunnel encapsulation settings are applied to static routes that reference the next-hop-group.

Example: Configure static next-hops

The following example configures two next hops for a static route. 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.

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default static next-hop *
    network-instance default {
        static {
            next-hop 1 {
                ip-address 192.0.2.22
            }
            next-hop 2 {
                ip-address 192.0.2.45
                resolve true
            }
        }
    }
```

Example: Add static next-hops to a static next-hop-group

The following example adds the previously defined next-hops to a static next-hop-group. If you are configuring multiple next-hop groups, you can reference the same next-hop object if the forwarding details are the same.

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default static next-hop-group static-ipv4-grp
    network-instance default {
        static {
            next-hop-group static-ipv4-grp {
                next-hop 1 {
                }
                next-hop 2 {
                }
            }
        }
    }
```

Example: Configure an IPv4 static route

The following example configures an IPv4 static route prefix to point to a static next-hop group.

```
--{ + candidate shared default }--[  ]--
A:root@srl1# info with-context network-instance default static-routes route 192.168.1.0/24
    network-instance default {
        static-routes {
            route 192.168.1.0/24 {
                admin-state enable
                metric 1
                preference 5
                static-next-hop-group static-ipv4-grp
            }
        }
    }
```

Example: Configure Generic UDP Encapsulation (GUE) tunnel settings for a static route

The following example configures GUE tunnel settings (`encap-header type udp-v4`) for a next-hop, adds the next-hop to a static next-hop-group, and configures an IPv4 static route prefix to point to the static next-hop group. UDPv4 tunnel encapsulation is only configurable for a network-instance of type default.

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default static next-hop 3
    network-instance default {
        static {
            next-hop 3 {
                encap-header 1 {
                    type udp-v4
                    udp-v4 {
                        source-ip 20.20.20.20
                        destination-ip 10.10.10.10
                        destination-udp-port 6080
                        ip-ttl 255
                    }
                }
            }
        }
```

```
    }
```

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default static next-hop-group gue-encap
    network-instance default {
        static {
            next-hop-group gue-encap {
                next-hop 3 {
                }
            }
        }
    }
```

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default static-routes route 192.168.2.0/24
    network-instance default {
        static-routes {
            route 192.168.2.0/24 {
                admin-state enable
                static-next-hop-group gue-encap
            }
        }
    }
```

9.3.3 Static route tags

A static route can have either:

- a configured association with a `tag-set` defined under `routing-policy.tag-set` with the tag-set representing a grouping of 1 or 2 administrative tag values; or
- a configured (singular) `tag-value`

Regardless of the method used to bind a tag (or tags) to a static route, the operator decides the meaning of each tag value. The tag value(s) that are associated with a static route can be used to match the static route in the following types of policies:

- BGP export policy
- VRF export policy
- BGP route-table import policy
- OSPF export policy
- ISIS export policy
- Network instance route leaking export policy
- Network instance route leaking import policy
- Any table-connection import-policy with `static` as a source protocol

9.3.3.1 Binding tags to static routes

Procedure

You can associate a static route with a tag-set, which is defined under routing-policy, or a single tag-value.

Example: Associate a static route with a tag-set

The following example defines a tag-set, then associates it with a static route.

```
--{ * candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        tag-set ts1 {
            tag-value [
                2
                4
            ]
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default static-routes
    network-instance default {
        static-routes {
            route 1.1.1.1/32 {
                tag-set ts1
            }
        }
    }
```

Example: Associate a static route with a tag-value

The following example associates a single tag-value with a static route.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default static-routes
    network-instance default {
        static-routes {
            route 10.10.10.10/32 {
                tag-value 5
            }
        }
    }
```

9.3.4 Configuring failure detection for static routes

Procedure

You can use BFD 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.

You enable BFD 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 with-context next-hop-groups
    next-hop-groups {
        group static-ipv4-grp {
            admin-state enable
            nexthop 1 {
                failure-detection {
                    enable-bfd {
                        local-address 192.0.2.1
                    }
                }
            }
        }
    }
```

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 "Bidirectional Forwarding Detection" in the *SR Linux OAM and Diagnostics Guide*.

9.4 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 process 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.

9.4.1 Configuring aggregate routes

Procedure

To specify an aggregate route, you configure the aggregator address setting, which 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 with-context aggregate-routes
    aggregate-routes {
        route 192.0.2.0/24 {
            aggregator {
                address 192.168.0.1
            }
            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.

9.5 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 must 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 13: 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 13: Route preference defaults by route type

Route Type	Preference	Configurable
Direct attached	0	No
Static routes	5	Yes

Route Type	Preference	Configurable
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
Aggregate routes	130	No

Note:

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

9.6 IP tunnel decapsulation groups

Within the default network-instance, you can configure IP tunnel decapsulation groups. In an IP tunnel decapsulation group, you specify a termination-subnet and an allowed-payload type. An IP/GRE packet received on any routed subinterface of the default network instance that matches the termination-subnet is decapsulated. If the inner header of the decapsulated packet indicates the packet is an allowed-payload type, it is forwarded according to the packet's inner header. If the decapsulated packet is not an allowed-payload type, it is dropped and not forwarded further.

A received IP/GRE packet that does not match the termination subnet in an IP tunnel decapsulation group is forwarded based on IP route lookup. If the longest matching route is a host route, then the packet is dropped unless it is explicitly allowed by a non-default CPM-filter policy.

Note:

- IP tunnel decapsulation groups are supported only on the default network instance.
- The termination-subnet lookup for a IP/GRE packet is done before IP longest prefix match lookup, so decapsulation occurs for a packet matching the termination-subnet even if the packet's outer IP address matches a local interface IP address or IP route in the FIB.
- Decapsulation occurs regardless of whether the GRE header specifies version 0 or 1.
- Decapsulation does not occur if the GRE version 0 header includes a key or sequence number. In this case, the packet is forwarded based on IP route lookup on the outer IP DA. If the longest matching route is a host route, the packet is dropped unless it is explicitly allowed by a non-default CPM-filter policy.

9.6.1 Configuring an IP tunnel decapsulation group

Procedure

To configure an IP tunnel decapsulation group, specify a termination-subnet and an allowed-payload type.

Example

The following example configures an IP tunnel decapsulation group in the default network instance. The termination-subnet is 192.168.1.0/24, and the allowed-payload type is MPLS. IP/GRE packets that match this subnet are decapsulated. If the payload of the decapsulated packet is MPLS, it is forwarded based on its inner header; otherwise, the decapsulated packet is dropped.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default ip-tunnel-decapsulation
    network-instance default {
        ip-tunnel-decapsulation {
            group dcgrp1 {
                allowed-payloads [
                    mpls
                ]
                termination-subnet 192.168.1.0/24 {
                }
            }
        }
    }
```

9.7 Displaying network-instance status

Procedure

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

Example: Display information about all network-instances

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: Display information about one network-instance

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: Display information about attached interfaces

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.

Example: Display a summary of IPv4 unicast routes for a network-instance

The following example displays a summary of IPv4 unicast routes for the mgmt network-instance:

```
--{ show }--[  ]--
# show network-instance mgmt ipv4 route summary
============================================================
IPv4-unicast route summary for ip-vrf network-instance: mgmt
------------------------------------------------------------
Route Type         Total Routes
------------------------------------------------------------
dhcp               1
host               2
linux              1
local              1
------------------------------------------------------------
Prefix Length      Active Routes
------------------------------------------------------------
/0                 1
/24                1
/32                2
------------------------------------------------------------
Total (active and inactive) routes  : 5
Total active routes                 : 4
Imported leaked routes              : 0
```

```
ECMP routes                          : 0
  Resilient ECMP routes              : 0
  Dynamic load-balancing ECMP routes: 0
FIB failed routes                    : 0
FIB suppressed routes                : 0
Routes with per-prefix statistics    : 0
```

Example: Display a full report of IPv4 unicast routes for a network-instance

The following example displays a full report for non-host IPv4 unicast routes for the mgmt network-instance:

```
--{ show }--[  ]--
# show network-instance mgmt ipv4 route all
=============================================================
IPv4-unicast route table for ip-vrf network-instance: mgmt
-------------------------------------------------------------
Flags: > (best), * (unviable), ! (failed)
     : L (leaked route from another network-instance)
     : B (backup NHG active and displayed)
     : S (statistics supported)
     : D (dynamic LB), R (resilient LB)
-------------------------------------------------------------------------------
Prefix              Route Type   Metric   Pref    Flags    Next-Hop(s)
-------------------------------------------------------------------------------
0.0.0.0/0           dhcp         0        5       >        172.20.20.1(mgmt0.0)
172.20.20.0/24      linux        0        5                172.20.20.0(mgmt0.0)
172.20.20.0/24      local        0        0       >        172.20.20.3(mgmt0.0)
```

Example: Display a route-table report for IPv4 routes of type host

The following example displays a route-table report for IPv4 routes of type host:

```
--{ show }--[  ]--
# show network-instance mgmt ipv4 route type host
============================================================
IPv4-unicast route table for ip-vrf network-instance: mgmt
------------------------------------------------------------
Prefix              Type
------------------------------------------------------------
172.20.20.3/32      extract
172.20.20.255/32    broadcast
```

Example: Display interface information using an interface reference

If you configure an interface reference for an interface (see Configuring interface references) in the network-instance configuration, you can display information about the interface using the interface reference name.

The following example configures network-instance "black" to use the interface reference "red" to refer to interface ethernet-1/10, subinterface 1.

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance black
    network-instance black {
        interface red {
            interface-ref {
                interface ethernet-1/10
                subinterface 1
            }
        }
```

```
    }
```

In the context of network instance black, you can use the name red to refer to interface ethernet-1/10, subinterface 1. For example:

```
--{ + running }--[ network-instance black ]--
# show interfaces red
================================================================================
Net instance    : black
Interface       : ethernet-1/10.1 (red)
Type            : routed
Oper state      : down
Oper down reason: subif-down
Ip mtu          : 1500
================================================================================
```

9.8 The mac-vrf network-instance

The network-instance type **mac-vrf** is associated with a network-instance of type **default** or **ip-vrf** via an 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.

9.8.1 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 (eXtensible Data Path (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

9.8.2 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.

9.8.2.1 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. When exceeding the threshold, the system holds on to the prior local destination of the MAC and executes an action.

9.8.2.2 MAC duplication actions

The action taken when 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**

  On detection of a duplicate mac on the subinterface, the mac is blackholed.

- **stop-learning**

  On detection of a duplicate mac on the subinterface, the MAC address is no longer relearned on this or any subinterface. This is the default action for a mac-vrf network instance.

- **use-net-instance-action**

  (Only available for subinterfaces) Use the action specified for the mac-vrf network instance. This is the default action for a subinterface.

9.8.2.3 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 network-instance bridge-table mac-duplication duplicate-entries** CLI command.

9.8.2.4 Configurable hold-down-time

The **info from state network-instance bridge-table 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.

9.8.3 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.

9.8.3.1 Deleting entries from the bridge table

Procedure

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: Clear MAC entries with a blackhole destination

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-macs-type
 blackhole-only
```

Example: Clear a specified MAC entry

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 learnt-entries mac
 00:00:5e:00:53:01 delete-mac
```

Example: Clear duplicate MAC entries for a subinterface

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
```

9.8.4 The mac-vrf network-instance type

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 three 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 type is associated with a bridged interface and an IRB interface.

```
--{ candidate shared default }--[  ]--
# info with-context 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-ref {
                interface ethernet-1/1
                subinterface 1
            }
        }
        interface irb1.1 {
            interface-ref {
                interface irb1
                subinterface 1
            }
        }
        bridge-table {
            mac-limit {
                maximum-entries 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
                }
            }
        }
    }
```

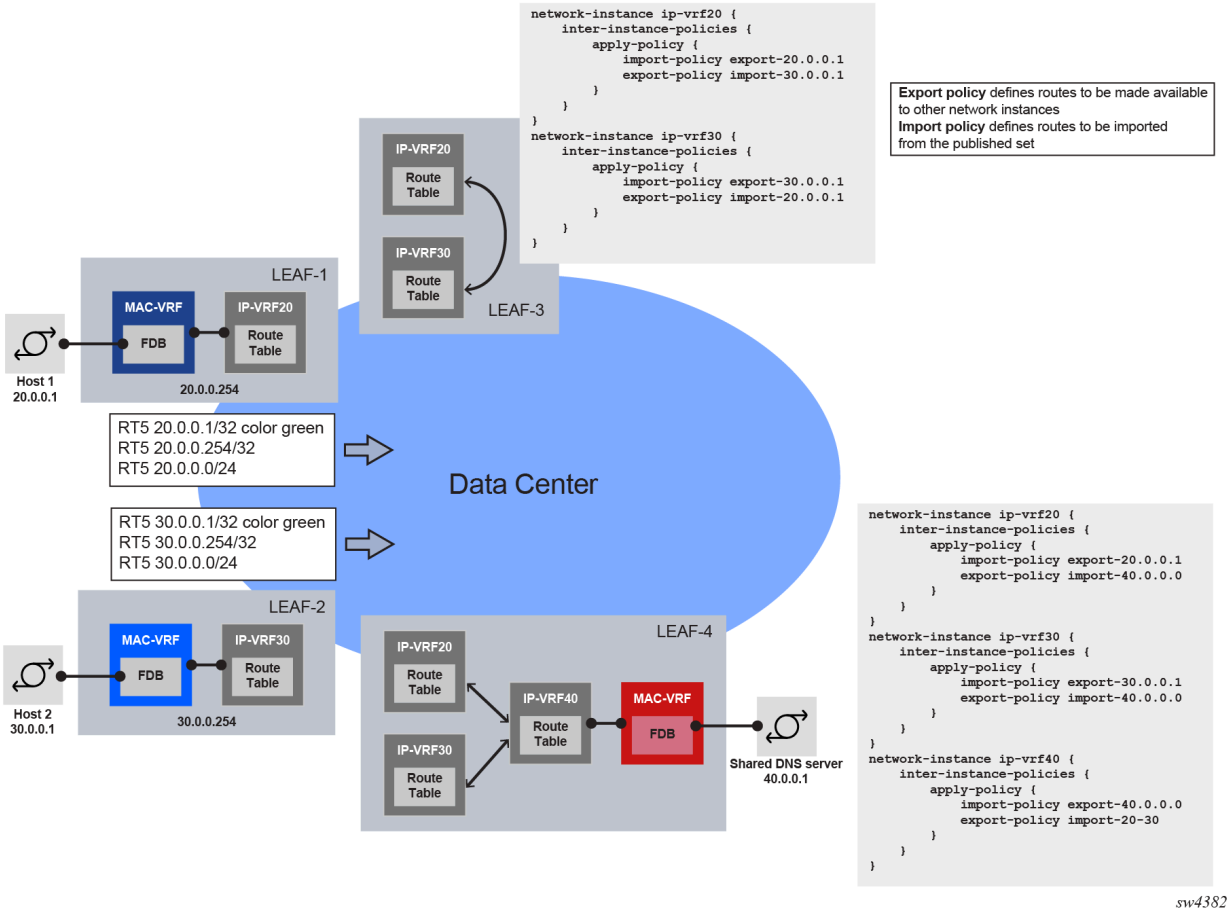
9.9 Network-instance route leaking

SR Linux supports route leaking from one network-instance to another. An active/best IP route from an origin network-instance can be leaked into any number of destination network-instances.

The routes are leaked from an origin network-instance to destination network-instances based on import and export policies. The policies specify which routes are leaked from the origin network-instance to destination network-instances, as well as which routes can be accepted by a network-instance from other network instances. Import policy matching can be based on prefix, family, and protocol. The leaked routes are published with full forwarding information (next hop).

Network-instance route leaking is typically used to allow leafs attached to multiple tenants to share services (for example, DNS or NTP servers, common software repositories) and support leafs that provide hub functionality to spoke leafs. Figure 3: Network-instance route leaking shows an example configuration.

Figure 3: Network-instance route leaking



In this example, Leaf-4 has two customer IP-VRF network-instances: IP-VRF20 and IP-VRF30. Network-instance IP-VRF40 is used to connect the shared DNS server with IP address 40.0.0.1. An inter-instance export policy makes the shared DNS server leakable, and an inter-instance import policy imports 40.0.0.0 into IP-VRF20 and IP-VRF30. At this point, the prefix 40.0.0.0/24 can be advertised via EVPN and imported into Leaf-1 and Leaf-2, so that Host 1 and Host 2 have reachability to the DNS server. In the reverse direction, for traffic from the hosts to the DNS server, inter-instance import and export policies are configured so that the IP-VRF20 and IP-VRF30 prefixes are imported into IP-VRF40.

The hub-and-spoke functionality is shown on Leaf-3, where prefixes of IP-VRF20 and IP-VRF30 are leaked between each other, allowing connectivity between Host 1 and Host 2.

Configuration considerations for originating and terminating traffic

This feature does not prevent leaking of host routes. However, packets matching a local host route leaked from another network-instance are dropped by XDP CPM. For example, if a router is leaking a local

loopback address from IP-VRF-A into IP-VRF-B, an ICMP request packet from IP-VRF-B with a destination of the loopback address in IP-VRF-A is discarded.

Packets matching a non-local host address belonging to a local subnet leaked from another network-instance are dropped. For example, if a router is leaking local subnet 10.10.10.0/24 on subinterface-1 from IP-VRF-A into IP-VRF-B and 10.10.10.10 is a server connected to subinterface-1 in IP-VRF-A, an ICMP request packet coming from IP-VRF-B with destination 10.10.10.10 is discarded. As a workaround, subinterface-1 can be configured with **host-route populate dynamic datapath-programming true** so that 10.10.10.10 creates and programs an ARP/ND host route in IP-VRF-A. If the ARP/ND host route is leaked into IP-VRF-B, traffic to 10.10.10.10 is forwarded from IP-VRF-B to IP-VRF-A.

Another workaround is to configure a static-route for 10.10.10.10 with next hop 10.10.10.10, and leak it into IP-VRF-B. Overall, any route that creates an entry in the host table for IP-VRF-B provides a way to forward packets to a local subnet.

XDP CPM uses (and does not ignore) leaked routes when forwarding originated control/management traffic. If a leaked route is the best route to the destination, then traffic egresses via a subinterface of the origin-network-instance.

On 7220 IXR-D1, 7220 IXR-D2, 7220 IXR-D3, 7220 IXR-D4, and 7220 IXR-D5 switches, VXLAN traffic received in a non-default VRF is not terminated, even if the destination IP address matches a leaked host route from the default network-instance.

If the next hop of any non-leaked route can only be resolved by a leaked route, the non-leaked route is considered unresolved. For example, if the BGP next hop of an EVPN-VXLAN route received by the default network-instance can only be resolved by a leaked route from another network-instance, the BGP route is considered unresolved. This process ensures VXLAN traffic is not originated from a non-default VRF.

Tie breaking process for leaked routes

When a route is leaked from network-instance A to network-instance B, it is possible that network-instance B already has one or more leaked and, or unleaked routes for the same IP prefix. SR Linux breaks these ties using the following sequence of rules:

1. Lowest preference
2. Lowest metric
3. Lowest route-type (non-leaked has a lower value than leaked)
4. Lowest origin network-instance ID

A protocol in network-instance B only sees a route of the same protocol from another network-instance (that is, a leaked route) if it is for a different prefix. Therefore in a situation where there are leaked and unleaked routes for the same prefix, there is no opportunity to select the best one based on protocol-level details (for example, BGP best path selection).

9.9.1 Re-advertising leaked routes

Leaked routes originated by protocols running in a source network-instance can be re-advertised by protocols running in a destination network-instance.

To do this, you can configure the following routing policy match conditions that apply to leaked routes:

- `origin-network-instance`

This condition specifies the source network-instance where the route was originally learned or configured. If this match condition is omitted from a policy statement, the implied match is any network-instance.

- `network-instance-leaked-route`

  When set to `true`, this condition matches all leaked routes (that is, routes with an origin network-instance different from the local-context network-instance). When set to `false`, this condition matches all non-leaked routes (that is, with an origin network-instance the same as the local-context network-instance).

The `origin-network-instance` and `network-instance-leaked-route` match conditions can be combined with other match criteria to refine the selection criteria in a routing policy. See "Routing policies" in the *SR Linux Routing Protocols Guide*.

The following table summarizes how leaked routes are re-advertised by protocols in a destination network-instance.

Table 14: How leaked routes are re-advertised in the destination network-instance

Original protocol (source network-instance)	Re-advertising protocol (destination network-instance)	Re-advertisement
non-BGP and `bgp-vpn`	`bgp` `bgp-label`	Leaked routes are added to the unlabeled and labeled RIB automatically unless rejected by the relevant route-table-import policy. When imported into a BGP RIB, a leaked route determined by SR Linux to be the best route is advertised only if accepted by a peer export policy or table-connection policy.
`bgp` `bgp-label`	`bgp` `bgp-label`	Leaked routes are not added to the unlabeled + label RIB unless they are accepted by the relevant route-table-import policy. When imported into a BGP RIB, a leaked route determined by SR Linux to be the best route is advertised unless rejected by a peer export policy.
Any except `bgp-vpn`	`bgp-vpn`	Leaked routes are not advertised unless accepted by the network-instance export policy
`bgp-vpn`	`bgp-vpn`	Leaked routes are not advertised unless accepted by the network-instance export policy. Note that leaked IP-VPN routes are not re-advertisable as IP-VPN routes unless `allow-export` is configured. Similarly, leaked EVPN routes are not re-advertisable as EVPN routes unless `allow-export` is configured.
Any	`isis`	Leaked routes are not advertised unless accepted by the IS-IS export policy or table-connection policy (in the case of BGP→IS-IS).

Original protocol (source network-instance)	Re-advertising protocol (destination network-instance)	Re-advertisement
		Note that the table-connection policy in the destination network-instance does not have access to the original BGP attributes. By default, leaked routes are advertised on the level configured as the level-capability of the re-advertising protocol instance (irrespective of the up/down bit of the leaked route, which would have been derived from protocol operation in the source network-instance).
Any	`ospf`	Leaked routes are not advertised unless accepted by the OSPF export policy

Attributes applied to re-advertised routes

When the advertising protocol Y is different from the leaked route's protocol X, the construction of the re-advertised route requires no special handling; the TLVs and attributes of the re-advertised route are the same as they are for a non-leaked route of protocol X. However, when Y and X are the same protocol, the construction of the re-advertised route copies some protocol-specific attributes from the leaked route so that leaking does not create routing loops. The following table summarizes this behavior.

Table 15: Attributes applied to re-advertised routes

Original protocol (X)	Re-advertising protocol (Y)	Re-advertised route attributes
BGP	IGP	No different than if the BGP route is local. LSA/LSP encoding indicates that the prefix is "external".
BGP	BGP	Same as original, except NEXT_HOP is reset to SELF and MED is reset to ZERO. All other attributes are copied exactly. Note that no split-horizon processing applies. A leaked iBGP route can be advertised to an iBGP peer without any route-reflector configuration in the destination network-instance.
BGP	BGP-VPN	Same as original, except NEXT_HOP is reset to SELF and MED is reset to ZERO. All other attributes are copied exactly. Note that no split-horizon processing applies. A leaked iBGP route can be advertised to an iBGP peer without any route-reflector configuration in the destination network-instance.
IGP	IGP	LSA/LSP encoding indicates that the prefix is "external"
IGP	BGP or BGP-VPN	No different than if the IGP route is local

Re-advertised leaked routes effects on table-connections

Table-connections (see the *SR Linux Routing Protocols Guide*) are a method to manage redistribution of routes between protocols. When table-connections are configured in the source or destination network-instance for leaked routes, it has the following effects:

When table-connections are enabled in the source network-instance:

- The same route can be both leaked to another network-instance, and redistributed by a table-connection import policy to another protocol in the same source network-instance.
- Routes redistributed by a table-connection import policy cannot be leaked because the redistributed route is not programmed into the FIB.

When table-connections are enabled in the destination network-instance, an explicit table-connection is required to re-advertise the leaked route, unless the original protocol is the same as the re-advertising protocol. Note that `bgp`, `bgp-label`, and `bgp-vpn` are all different protocols in this context.

9.9.2 Matching BGP attributes in a route-leaking-export policy

The route-leaking export-policy in a network-instance can be one that matches BGP routes based on BGP-specific attributes.

If the policy referenced by the `network-instance.inter-instance-policies.apply-policy.export-policy` command contains any of the following BGP-specific match conditions, SR Linux uses the BGP attributes when evaluating the policy and determining whether a route should be leaked.

- BGP AS path attributes (configured with `routing-policy.policy.statement.match.bgp.as-path`)
- BGP standard communities (configured with `routing-policy.policy.statement.match.bgp.standard-community`)
- BGP extended communities (configured with `routing-policy.policy.statement.match.bgp.extended-community`)

For example, the following defines a community set, creates a routing policy that accepts routes matching this community set, and applies the routing policy as the export policy in a route-leaking configuration in the default network-instance:

```
--{ * candidate shared default }--[  ]--
# info with-context routing-policy standard-community-set st1
    routing-policy {
        standard-community-set st1 {
            member [
                65000:65000
            ]
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        policy st1-communities {
            statement 1 {
                match {
                    bgp {
                        standard-community-set st1
```

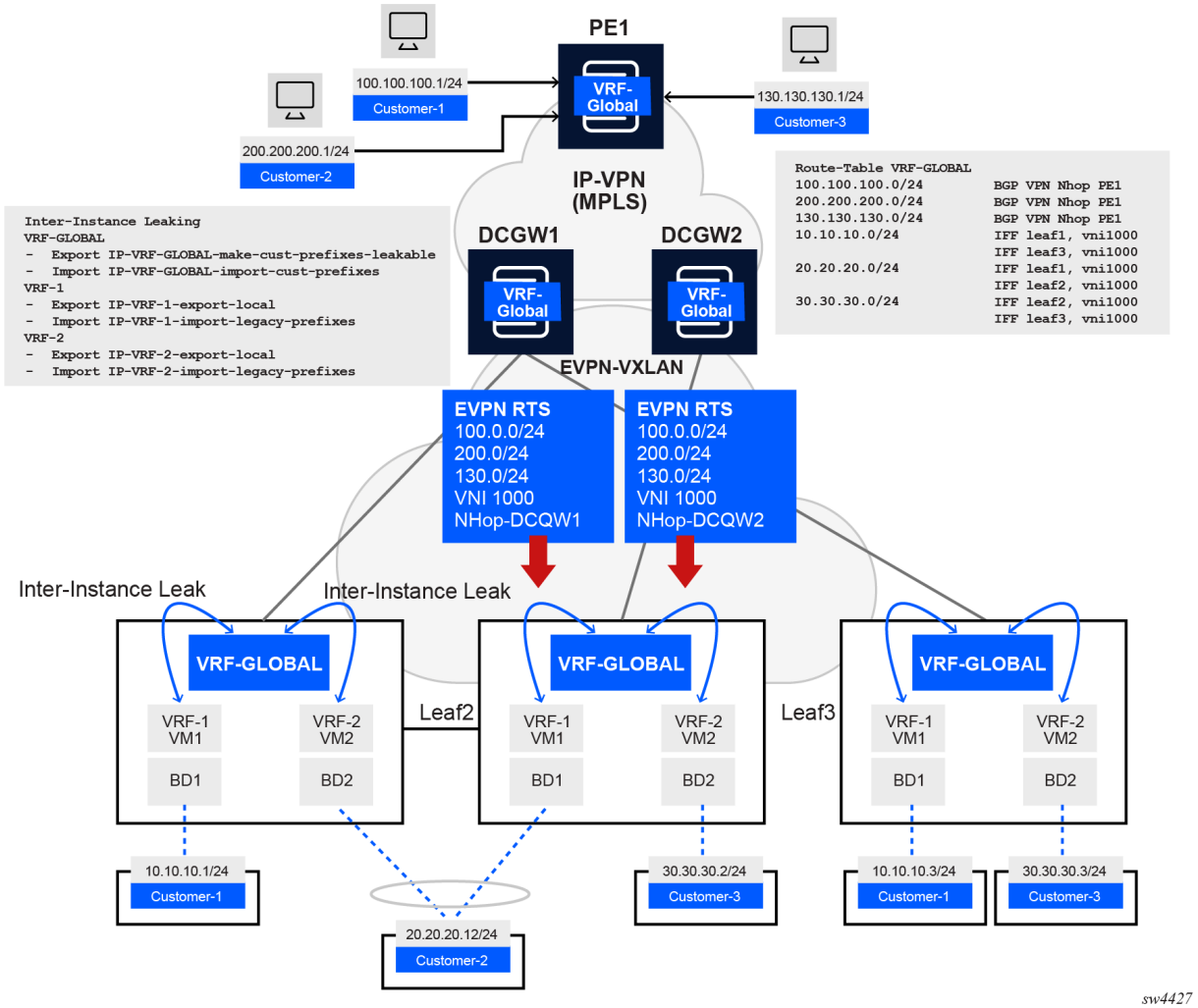
```
                    }
                }
                action {
                    policy-result accept
                }
            }
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default inter-instance-policies
    network-instance default {
        inter-instance-policies {
            apply-policy {
                export-policy [
                    st1-communities
                ]
            }
        }
    }
```

9.9.3 Network-instance route leaking configuration example

The following diagram shows a configuration for a hosting provider that uses route leaking on EVPN-VXLAN leaf switches.

Figure 4: Network-instance route leaking example



The data center network in this example uses SR Linux leaf switches and 7750 SR devices as data center (DC) gateways. Each customer is assigned an IP-VRF and a MAC-VRF on the leaf where their server is connected. For example, Customer-1 has a server connected to Leaf1 (10.10.10.1) and another server connected to Leaf3 (10.10.10.3). Both servers are attached to a bridged subinterface (no VLAN tagging) that is associated with MAC-VRF BD1. Regular EVPN-VXLAN connectivity happens for each customer with servers attached to multiple leaf switches.

Route leaking is necessary in this example because each customer also requires connectivity to some hosts that are connected to an IP-VPN network in the WAN, and this connectivity is not directly programmed in the IP-VRF of the customer.

The route-leaking example shown in Figure 4: Network-instance route leaking example shows the following:

- Customer-1, Customer-2, and Customer-3 hosts and prefixes are learned on the IP-VRF VRF-GLOBAL on PE1.

- PE1 advertises the three prefixes using IP-VPN control plane; DCGW1 and DCGW2 import those prefixes in VRF-GLOBAL as well.
- In addition to supporting IP-VPN, the VPRN service on the 7750 DCGWs is configured with an EVPN-VXLAN instance that is used to advertise the customer legacy WAN prefixes to the leaf switches, and to import the customer prefixes of all customers attached to the leaf nodes.
- A network-instance of type IP-VRF called VRF-GLOBAL is also configured on the three leaf switches.

Note: Some 7220 IXR and 7250 IXR platforms have an issue where leaked packets sent to a local leaked subnet are blackholed, as described in Configuration considerations for originating and terminating traffic. In the preceding Figure 4: Network-instance route leaking example, this means traffic from VRF-GLOBAL to VRF-1, with the destination of a local subnet in VRF-1, is blackholed. As a workaround, the VRF-1 IRB subinterface can be configured with `host-route.populate[dynamic]` and `host-route.populate[dynamic].datapath-programming true`, so that active hosts create ARP/ND host routes, and they are leaked into VRF-GLOBAL. The commands create a host in the VRF-GLOBAL host table, and traffic can successfully be forwarded to the ARP/ND learned hosts in VRF-1.

The following sections show the relevant configuration and state information for the PE and Leaf devices in the route leaking example shown in Figure 4: Network-instance route leaking example.

- PE1 configuration and route table
- DCGW1 configuration and route table
- Leaf1 customer VRF configuration
- Leaf1 VRF-GLOBAL configuration
- Leaf1 routing policy configuration
- Leaf1 IP-VRF route-tables
- Leaf1 route-tables state

PE1 configuration and route table

In the route leaking example shown in Figure 4: Network-instance route leaking example, PE1 is a 7750 SR WAN PE device attached to VRF-GLOBAL, where the three hosts are connected. The following example shows the configuration and route table for PE1.

```
// VRF-GLOBAL config on PE1

[ex:/configure service vprn "IP-VRF-GLOBAL"]
A:admin@pe1# info
    admin-state enable
    service-id 1000
    customer "1"
    bgp-ipvpn {
        mpls {
            admin-state enable
            route-distinguisher "10.0.0.5:1000"
            vrf-target {
                community "target:64500:1000"
            }
            auto-bind-tunnel {
                resolution any
            }
        }
    }
```

```
    interface "local-100" {
        ipv4 {
            admin-state enable
            primary {
                address 100.100.100.254
                prefix-length 24
            }
        }
        sap 1/1/c1/1:0.* {
        }
    }
    interface "local-200" {
        ipv4 {
            admin-state enable
            primary {
                address 200.200.200.254
                prefix-length 24
            }
        }
        sap 1/1/c2/1:0.* {
        }
    }
    interface "local-300" {
        ipv4 {
            admin-state enable
            primary {
                address 130.130.130.254
                prefix-length 24
            }
        }
        sap 1/1/c3/1:3.0 {
        }
    }

// The IP-VRF-GLOBAL route-table shows the routes 100.100.100.0/24 (customer-1), 200.200.200.0/
24 (customer-2) and 130.130.130.0/24 (customer-3)
// PE1 also receives via IPVPN the corresponding prefixes from Leaf-1 so that each customer can
 have communication to/from the legacy hosts

[ex:/configure service vprn "IP-VRF-GLOBAL"]
A:admin@pe1# /show router "1000" route-table

===============================================================================
Route Table (Service: 1000)
===============================================================================
Dest Prefix[Flags]                              Type    Proto     Age        Pref
      Next Hop[Interface Name]                                    Metric
-------------------------------------------------------------------------------
10.10.10.0/24                                   Remote  BGP VPN   01h20m56s  170
       10.0.0.4 (tunneled:SR-ISIS:524290)                           10
20.20.20.0/24                                   Remote  BGP VPN   01h18m59s  170
       10.0.0.4 (tunneled:SR-ISIS:524290)                           10
100.100.100.0/24                                Local   Local     07h55m03s  0
       local-100                                                    0
130.130.130.0/24                                Local   Local     07h55m03s  0
       local-300                                                    0
200.200.200.0/24                                Local   Local     07h55m03s  0
       local-200                                                    0
-------------------------------------------------------------------------------
No. of Routes: 5
Flags: n = Number of times nexthop is repeated
       B = BGP backup route available
       L = LFA nexthop available
       S = Sticky ECMP requested
```

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

DCGW1 configuration and route table

DCGW1 is also attached to VRF-GLOBAL and provides a gateway between IP-VPN and EVPN-VXLAN. The following example shows the configuration and route table for DCGW1.

```
// VRF-GLOBAL config on DCGW1

[ex:/configure service vprn "IP-VRF-GLOBAL"]
A:admin@dcgw1# info
    admin-state enable
    service-id 1000
    customer "1"
    bgp-ipvpn {
        mpls {
            admin-state enable
            route-distinguisher "10.0.0.4:1000"
            vrf-target {
                community "target:64500:1000"
            }
            auto-bind-tunnel {
                resolution any
            }
        }
    }
    interface "sdb-1001" {
        vpls "sdb-1001" {
            evpn-tunnel {
                ipv6-gateway-address mac
            }
        }
        ipv6 {
        }
    }

// DCGW1 learns the legacy prefixes via IPVPN and the leaf prefixes from EVPN IFF (EVPN VXLAN)

A:admin@dcgw1# /show router "1000" route-table
===============================================================================
Route Table (Service: 1000)
===============================================================================
Dest Prefix[Flags]                            Type    Proto     Age        Pref
      Next Hop[Interface Name]                                    Metric
-------------------------------------------------------------------------------
10.10.10.0/24                                 Remote  EVPN-IFF  00h00m13s  169
       sdb-1001 (ET-1a:8b:08:ff:00:00)                                0
20.20.20.0/24                                 Remote  EVPN-IFF  00h00m13s  169
       sdb-1001 (ET-1a:8b:08:ff:00:00)                                0
30.30.30.0/24                                 Remote  EVPN-IFF  00h00m13s  169
       sdb-1001 (ET-1a:b2:09:ff:00:00)                                0
100.100.100.0/24                              Remote  BGP VPN   00h00m13s  170
       10.0.0.5 (tunneled:SR-ISIS:524290)                             10
130.130.130.0/24                              Remote  BGP VPN   00h00m13s  170
       10.0.0.5 (tunneled:SR-ISIS:524290)                             10
200.200.200.0/24                              Remote  BGP VPN   00h00m13s  170
       10.0.0.5 (tunneled:SR-ISIS:524290)                             10
-------------------------------------------------------------------------------
No. of Routes: 6
Flags: n = Number of times nexthop is repeated
       B = BGP backup route available
       L = LFA nexthop available
       S = Sticky ECMP requested
```

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

Leaf1 customer VRF configuration

Leaf1 in the route leaking example shown in Figure 4: Network-instance route leaking example is an SR Linux device attached to customer IP-VRFs VRF-1 and VRF-2 and their corresponding MAC-VRFs. The following is a configuration on Leaf1 for the IP-VRFs and MAC-VRFs.

```
// Relevant config for VRF-1

--{ + candidate shared default }--[ network-instance * ]--
A:leaf1# info
    network-instance BD1 {
        type mac-vrf

        interface ethernet-1/1.1 {
            interface-ref {
                interface ethernet-1/1
                subinterface 1
            }
        }
        interface irb0.1 {
            interface-ref {
                interface irb0
                subinterface 1
            }
        }
        vxlan-interface vxlan1.11 {
        }
        protocols {
            bgp-evpn {
                bgp-instance 1 {
                    admin-state enable
                    vxlan-interface vxlan1.11
                    evi 11
                    ecmp 8
                    routes {
                        bridge-table {
                            mac-ip {
                                advertise-arp-nd-only-with-mac-table-entry true
                            }
                        }
                    }
                }
            }
            bgp-vpn {
                bgp-instance 1 {
                    route-target {
                        export-rt target:100:11
                        import-rt target:100:11
                    }
                }
            }
        }
    }
    network-instance IP-VRF-1 {
        type ip-vrf
        inter-instance-policies {
            apply-policy {
                import-policy IP-VRF-1-import-cust-1
                export-policy IP-VRF-export-local
            }
```

```
        }
        interface irb0.1 {
        }
        interface lo1.11 {
        }
        vxlan-interface vxlan1.1 {
        }
        protocols {
            bgp-evpn {
                bgp-instance 1 {
                    admin-state enable
                    vxlan-interface vxlan1.1
                    evi 1
                    ecmp 8
                }
            }
            bgp-vpn {
                bgp-instance 1 {
                    route-target {
                        export-rt target:100:1
                        import-rt target:100:1
                    }
                }
            }
        }
    }
// ARP/ND - host-route datapath-programming under subinterface

--{ + candidate shared default }--[ interface ethernet-1/3 subinterface 1 ]--
A:Dut-B# info
    admin-state enable
    ipv4 {
        admin-state enable
        address 2.3.0.1/30 {
        }
        arp {
            host-route {
                populate dynamic {
                    datapath-programming true
                }
            }
        }
    }
    ipv6 {
        admin-state enable
        address 3ffe:2:3::1/124 {
        }
        neighbor-discovery {
            host-route {
                populate dynamic {
                    datapath-programming true
                }
            }
        }
    }
// Relevant config for VRF-2

--{ + candidate shared default }--[ network-instance * ]--
A:leaf1# info
    network-instance BD2 {
        type mac-vrf
        interface irb0.2 {
        }
        interface lag1.1 {
```

```
        }
        vxlan-interface vxlan1.22 {
        }
        protocols {
            bgp-evpn {
                bgp-instance 1 {
                    admin-state enable
                    vxlan-interface vxlan1.22
                    evi 22
                    ecmp 8
                    routes {
                        bridge-table {
                            mac-ip {
                                advertise-arp-nd-only-with-mac-table-entry true
                            }
                        }
                    }
                }
            }
            bgp-vpn {
                bgp-instance 1 {
                    route-target {
                        export-rt target:100:22
                        import-rt target:100:22
                    }
                }
            }
        }
    }
    network-instance IP-VRF-2 {
        type ip-vrf
        inter-instance-policies {
            apply-policy {
                import-policy IP-VRF-2-import-cust-2
                export-policy IP-VRF-export-local
            }
        }
        interface irb0.2 {
        }
        interface lo1.12 {
        }
        vxlan-interface vxlan1.2 {
        }
        protocols {
            bgp-evpn {
                bgp-instance 1 {
                    admin-state enable
                    vxlan-interface vxlan1.2
                    evi 2
                    ecmp 8
                }
            }
            bgp-vpn {
                bgp-instance 1 {
                    route-target {
                        export-rt target:100:2
                        import-rt target:100:2
                    }
                }
            }
        }
    }
```

Leaf1 VRF-GLOBAL configuration

SR Linux Leaf1, as well as the other leaf switches, must be configured with VRF-GLOBAL so that the legacy prefixes for all customers are imported. The following is a configuration for VRF-GLOBAL on Leaf1.

```
--{ + candidate shared default }--[ network-instance IP-VRF-GLOBAL ]--
A:leaf1# info
    type ip-vrf
    admin-state enable
    inter-instance-policies {
        apply-policy {
            import-policy IP-VRF-GLOBAL-import-cust-prefixes
            export-policy IP-VRF-GLOBAL-make-cust-prefixes-leakable
        }
    }
    vxlan-interface vxlan1.1000 {
    }
    protocols {
        bgp-evpn {
            bgp-instance 1 {
                admin-state enable
                vxlan-interface vxlan1.1000
                evi 1000
                ecmp 8
            }
        }
        bgp-vpn {
            bgp-instance 1 {
                route-target {
                    export-rt target:100:1000
                    import-rt target:100:1000
                }
            }
        }
    }
```

Leaf1 routing policy configuration

In the route leaking example shown in Figure 4: Network-instance route leaking example, route leaking policies are required to guarantee communication between servers in the DC and hosts in the legacy IP-VPN network.

VRF-GLOBAL requires the following inter-instance policies:

- `export policy IP-VRF-GLOBAL-make-cust-prefixes-leakable`
  This policy makes the legacy prefixes 100.100.100.0/24, 200.200.200.0/24 and 130.130.130.0/24 in the VRF-GLOBAL route-table leakable. The policy matches on and accepts protocol bgp-evpn and prefix-set routes and rejects everything else to ensure no other routes are made available for leaking. Leaked routes are ignored by the policy.
- `import policy IP-VRF-GLOBAL-import-cust-prefixes`
  This policy imports all prefixes that are made leakable into the VRF-GLOBAL route-table.

VRF-1 requires the following inter-instance policies:

- `export policy IP-VRF-export-local`
  This policy matches on protocol local and protocol arp-nd so that the local/arp-nd routes of VRF-1 are made available for leaking.
- `import policy IP-VRF-1-import-cust-1`

This policy matches on prefix-set customer-1 (which comprises all prefixes in 100.100.100.0/24) to accept only the routes of customer-1 and reject the rest.

The policies for VRF-2 are similar to those for VRF-1, only replacing the `prefix-set` with `prefix-set customer-2`, which covers all prefixes in 200.200.200.0/24.

The following are the policies configured on SR Linux Leaf1. Similar policies are configured on the other leaf devices.

```
--{ + candidate shared default }--[ routing-policy ]--
A:leaf1# info
// prefix sets defined:
    prefix-set cust-1 {
        prefix 100.100.0.0/16 mask-length-range 16..24 {
        }
    }
    prefix-set cust-2 {
        prefix 200.200.0.0/16 mask-length-range 16..24 {
        }
    }
    prefix-set cust-3 {
        prefix 130.130.0.0/16 mask-length-range 16..24 {
        }
    }
    community-set target:100:1000 {
        member [
            target:100:1000
        ]
    }
    policy IP-VRF-1-import-cust-1 {  // leaks into IP-VRF-1 only the prefix-set of customer 1
        default-action {
            policy-result reject
        }
        statement 1 {
            match {
                prefix-set cust-1
            }
            action {
                policy-result accept
            }
        }
    }
    policy IP-VRF-2-import-cust-2 { // leaks into IP-VRF-2 only the prefix-set of customer 1
        default-action {
            policy-result reject
        }
        statement 1 {
            match {
                prefix-set cust-2
            }
            action {
                policy-result accept
            }
        }
    }
    policy IP-VRF-GLOBAL-import-cust-prefixes {  // leaks into IP-VRF-GLOBAL all the prefixes
 made leakable
        default-action {
            policy-result accept
        }
    }
    policy IP-VRF-GLOBAL-make-cust-prefixes-leakable {
        default-action {
```

```
            policy-result reject
        }
        statement 10 {
            match {
                protocol bgp-evpn
            }
            action {
                policy-result accept
            }
        }
    }
    policy IP-VRF-export-local { // applied to IP-VRF-1 and IP-VRF-2, makes leakable the routes
that are either local or arp-nd
        default-action {
            policy-result reject
        }
        statement 10 {
            match {
                protocol local
            }
            action {
                policy-result accept
            }
        }
        statement 20 {
            match {
                protocol arp-nd
            }
            action {
                policy-result accept
            }
        }
    }
    policy import-local-cust {  // applied as import on the IP-VRF-GLOBAL, imports only the
legacy prefixes of the local customers
        default-action {
            policy-result reject
        }
        statement 10 {
            match {
                prefix-set cust-1
                bgp {
                    community-set target:100:1000
                }
            }
            action {
                policy-result accept
            }
        }
        statement 20 {
            match {
                prefix-set cust-2
                bgp {
                    community-set target:100:1000
                }
            }
            action {
                policy-result accept
            }
        }
    }
```

Leaf1 IP-VRF route-tables

Based on the preceding policies, the route tables in Leaf1 shows the leaked routes as follows:

```
-{ + candidate shared default }--[  ]--
A:leaf1# show network-instance IP-VRF-* route-table
-------------------------------------------------------------------------------------------------------------
IPv4 unicast route table of network instance IP-VRF-1
-------------------------------------------------------------------------------------------------------------
+------------------+---+---------+-------------+-------+------+----+---------------------+----------+
|      Prefix      | ID| Route   |     Route   | Active|Metric|Pref|   Next-hop          | Next-hop |
|                  |   |  Type   |     Owner   |       |      |    |    (Type)           |Interface |
+==================+===+=========+=============+=======+======+====+=====================+==========+
| 10.10.10.0/      | 0 | bgp-evpn| bgp_evpn_mgr| False | 0    | 170| 10.0.0.3/32         |          |
|                  |   |         |             |       |      |    |  (indirect/vxlan)   |          |
| 10.10.10.0/24    | 4 | local   | net_inst_mgr| True  | 0    | 0  | 10.10.10.254 (direct)| irb0.1   |
| 10.10.10.1/32    | 4 | arp-nd  | arp_nd_mgr  | True  | 0    | 1  | 10.10.10.1 (direct)  | irb0.1   |
| 10.10.10.254/32  | 4 | host    | net_inst_mgr| True  | 0    | 0  | None (extract)       | None     |
| 10.10.10.255/32  | 4 | host    | net_inst_mgr| True  | 0    | 0  | None (broadcast)     |          |
| 11.11.11.11/32   | 7 | host    | net_inst_mgr| True  | 0    | 0  | None (extract)       | None     |
| 31.31.31.31/32   | 0 | bgp-evpn| bgp_evpn_mgr| True  | 0    | 170| 10.0.0.3/32         |          |
|                  |   |         |             |       |      |    | (indirect/vxlan)    |          |
| 100.100.100.0/24 | 0 | bgp-evpn| bgp_evpn_mgr| True  | 0    | 170|                     |          |
+------------------+---+---------+-------------+-------+------+----+---------------------+----------+
-------------------------------------------------------------------------------------------------------------
IPv4 routes total
IPv4 prefixes with active routes     : 7
IPv4 prefixes with active ECMP routes: 0
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
IPv4 unicast route table of network instance IP-VRF-2
-------------------------------------------------------------------------------------------------------------
+------------------+---+---------+-------------+-------+------+----+---------------------+----------+
|      Prefix      | ID| Route   |     Route   | Active|Metric|Pref|   Next-hop          | Next-hop |
|                  |   |  Type   |     Owner   |       |      |    |    (Type)           |Interface |
+==================+===+=========+=============+=======+======+====+=====================+==========+
| 12.12.12.12/32   | 8 | host    | net_inst_mgr| True  | 0    | 0  | None (extract)       | None     |
| 20.20.20.0/24    | 0 | bgp-evpn| bgp_evpn_mgr| False | 0    | 170| 10.0.0.2/32         |          |
|                  |   |         |             |       |      |    | (indirect/vxlan)    |          |
| 20.20.20.0/24    | 5 | local   | net_inst_mgr| True  | 0    | 0  | 20.20.20.254 (direct)| irb0.2   |
| 20.20.20.12/32   | 0 | bgp-evpn| bgp_evpn_mgr| True  | 0    | 170| 10.0.0.2/32         |          |
|                  |   |         |             |       |      |    | (indirect/vxlan)    |          |
| 20.20.20.254/32  | 5 | host    | net_inst_mgr|       | 0    | 0  | None (extract)       | None     |
| 20.20.20.255/32  | 5 | host    | net_inst_mgr| True  | 0    | 0  | None (broadcast)     |          |
| 22.22.22.22/32   | 0 | bgp-evpn| bgp_evpn_mgr| True  | 0    | 170| 10.0.0.2/32         |          |
|                  |   |         |             |       |      |    | (indirect/vxlan)    |          |
| 200.200.200.0/24 | 0 | bgp-evpn| bgp_evpn_mgr| True  | 0    | 170|                     |          |
+------------------+---+---------+-------------+-------+------+----+---------------------+----------+
-------------------------------------------------------------------------------------------------------------
IPv4 routes total                    : 8
IPv4 prefixes with active routes     : 7
IPv4 prefixes with active ECMP routes: 0
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
IPv4 unicast route table of network instan
-------------------------------------------------------------------------------------------------------------
+------------------+---+---------+-------------+-------+------+----+------------------+----------+
|      Prefix      | ID| Route   |     Route   | Active|Metric|Pref|   Next-hop       | Next-hop |
|                  |   |  Type   |     Owner   |       |      |    |    (Type)        |Interface |
+==================+===+=========+=============+=======+======+====+==================+==========+
| 10.10.10.0/24    | 4 | local   | net_inst_mgr| True  | 0    | 0  |                  |          |
| 10.10.10.1/32    | 4 | arp-nd  | arp_nd_mgr  | True  | 0    | 1  |                  |          |
| 20.20.20.0/24    | 5 | local   | net_inst_mgr| True  | 0    | 0  |                  |          |
| 100.100.100.0/24 | 0 | bgp-evpn| bgp_evpn_mgr| True  | 0    | 170| 10.0.0.4/32      |          |
```

```
|                  |   |         |             |       |      |    | (indirect/vxlan) |          |
| 200.200.200.0/24 | 0 | bgp-evpn| bgp_evpn_mgr| True  | 0    | 170| 10.0.0.4/32      |          |
|                  |   |         |             |       |      |    | (indirect/vxlan) |          |
+------------------+---+---------+-------------+-------+------+----+------------------+----------+
-------------------------------------------------------------------------------------------------
IPv4 routes total                    : 5
IPv4 prefixes with active routes     : 5
IPv4 prefixes with active ECMP routes: 0
-------------------------------------------------------------------------------------------------
--{ + candidate shared default }--[  ]--
```

Leaf1 route-tables state

The `info from state` information shows the origin of the leaked routes:

```
-{ + candidate shared default }--[ network-instance IP-VRF-* ]--
A:leaf1# info from state route-table
    network-instance IP-VRF-1 {
        route-table {
            ipv4-unicast {
                route 10.10.10.0/24 id 0 route-type bgp-evpn route-owner bgp_evpn_mgr origin-
network-instance IP-VRF-1 {
                    leakable false
                    metric 0
                    preference 170
                    active false
                    last-app-update 2023-01-26T14:59:31.386Z
                    next-hop-group 538392877111
                    next-hop-group-network-instance IP-VRF-1
                    resilient-hash false
                    fib-programming {
                        suppressed false
                        last-successful-operation-type modify
                        last-successful-operation-timestamp 2023-01-26T14:59:31.387Z
                        pending-operation-type none
                        last-failed-operation-type none
                    }
                }
<snip>
                route 100.100.100.0/24 id 0 route-type bgp-evpn route-owner bgp_evpn_mgr
 origin-network-instance IP-VRF-GLOBAL {
                    leakable true
                    metric 0
                    preference 170
                    active true
                    last-app-update 2023-01-26T14:59:31.386Z
                    next-hop-group 538392877123
                    next-hop-group-network-instance IP-VRF-GLOBAL
                    resilient-hash false
                    target-network-instances [
                        IP-VRF-1
                    ]
                    fib-programming {
                        suppressed false
                        last-successful-operation-type modify
                        last-successful-operation-timestamp 2023-01-26T14:59:31.397Z
                        pending-operation-type none
                        last-failed-operation-type none
                    }
                }
                statistics {
                    active-routes 7
                    active-routes-with-ecmp 0
```

```
                    resilient-hash-routes 0
                    fib-failed-routes 0
                    total-routes 8
                }
                route-summary {
                    route-type arp-nd {
                        active-routes 1
                    }
                    route-type bgp-evpn {
                        active-routes 3
                    }
                    route-type host {
                        active-routes 3
                    }
                    route-type local {
                        active-routes 1
                    }
                }
            }
            ipv6-unicast {
            }
            next-hop-group 538392877076 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
                    next-hop 538392877074
                    resolved not-applicable
                }
            }
            next-hop-group 538392877085 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
                    next-hop 538392877081
                    resolved not-applicable
                }
            }
            next-hop-group 538392877086 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
                    next-hop 538392877082
                    resolved not-applicable
                }
            }
            next-hop-group 538392877087 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
```

```
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
                    next-hop 3
                    resolved not-applicable
                }
            }
            next-hop-group 538392877111 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
                    next-hop 538392877104
                    resolved true
                }
            }
            next-hop-group 538392877120 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
                    next-hop 538392877111
                    resolved not-applicable
                }
            }
            next-hop 3 {
                type broadcast
            }
            next-hop 538392877074 {
                type extract
            }
            next-hop 538392877081 {
                type direct
                ip-address 10.10.10.254
                subinterface irb0.1
            }
            next-hop 538392877082 {
                type extract
            }
            next-hop 538392877104 {
                type indirect
                ip-address 10.0.0.3
                resolving-tunnel {
                    ip-prefix 10.0.0.3/32
                    tunnel-type vxlan
                    tunnel-owner vxlan_mgr
                }
                vxlan {
                    vni 1
                    source-mac 1A:8B:08:FF:00:00
                    destination-mac 1A:4D:0A:FF:00:00
                }
            }
            next-hop 538392877111 {
```

```
                type direct
                ip-address 10.10.10.1
                subinterface irb0.1
            }
        }
    }
    network-instance IP-VRF-2 {
        route-table {
            ipv4-unicast {
                route 12.12.12.12/32 id 8 route-type host route-owner net_inst_mgr origin-
network-instance IP-VRF-2 {
                    leakable false
                    metric 0
                    preference 0
                    active true
                    last-app-update 2023-01-26T14:59:31.384Z
                    next-hop-group 538392877077
                    next-hop-group-network-instance IP-VRF-2
                    resilient-hash false
                    fib-programming {
                        suppressed false
                        last-successful-operation-type modify
                        last-successful-operation-timestamp 2023-01-26T14:59:31.387Z
                        pending-operation-type none
                        last-failed-operation-type none
                    }
                }
<snip>
                route 200.200.200.0/24 id 0 route-type bgp-evpn route-owner bgp_evpn_mgr
 origin-network-instance IP-VRF-GLOBAL {
                    leakable false
                    metric 0
                    preference 170
                    active true
                    last-app-update 2023-01-26T14:59:31.386Z
                    next-hop-group 538392877123
                    next-hop-group-network-instance IP-VRF-GLOBAL
                    resilient-hash false
                    fib-programming {
                        suppressed false
                        last-successful-operation-type modify
                        last-successful-operation-timestamp 2023-01-26T14:59:31.387Z
                        pending-operation-type none
                        last-failed-operation-type none
                    }
                }
                statistics {
                    active-routes 7
                    active-routes-with-ecmp 0
                    resilient-hash-routes 0
                    fib-failed-routes 0
                    total-routes 8
                }
                route-summary {
                    route-type bgp-evpn {
                        active-routes 4
                    }
                    route-type host {
                        active-routes 3
                    }
                    route-type local {
                        active-routes 1
                    }
                }
```

```
            }
            ipv6-unicast {
            }
            next-hop-group 538392877077 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
                    next-hop 538392877075
                    resolved not-applicable
                }
            }
            next-hop-group 538392877082 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
                    next-hop 538392877079
                    resolved not-applicable
                }
            }
            next-hop-group 538392877083 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
                    next-hop 538392877080
                    resolved not-applicable
                }
            }
            next-hop-group 538392877084 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
                    next-hop 3
                    resolved not-applicable
                }
            }
            next-hop-group 538392877105 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
```

```
                    next-hop 538392877098
                    resolved true
                }
            }
            next-hop 3 {
                type broadcast
            }
            next-hop 538392877075 {
                type extract
            }
            next-hop 538392877079 {
                type direct
                ip-address 20.20.20.254
                subinterface irb0.2
            }
            next-hop 538392877080 {
                type extract
            }
            next-hop 538392877098 {
                type indirect
                ip-address 10.0.0.2
                resolving-tunnel {
                    ip-prefix 10.0.0.2/32
                    tunnel-type vxlan
                    tunnel-owner vxlan_mgr
                }
                vxlan {
                    vni 2
                    source-mac 1A:8B:08:FF:00:00
                    destination-mac 1A:B2:09:FF:00:00
                }
            }
        }
    }
    network-instance IP-VRF-GLOBAL {
        route-table {
            ipv4-unicast {
                route 10.10.10.0/24 id 4 route-type local route-owner net_inst_mgr origin-
network-instance IP-VRF-1 {
                    leakable false
                    metric 0
                    preference 0
                    active true
                    last-app-update 2023-01-26T14:59:31.384Z
                    next-hop-group 538392877085
                    next-hop-group-network-instance IP-VRF-1
                    resilient-hash false
                    fib-programming {
                        suppressed false
                        last-successful-operation-type modify
                        last-successful-operation-timestamp 2023-01-26T14:59:31.387Z
                        pending-operation-type none
                        last-failed-operation-type none
                    }
                }
                route 10.10.10.1/32 id 4 route-type arp-nd route-owner arp_nd_mgr origin-
network-instance IP-VRF-1 {
                    leakable false
                    metric 0
                    preference 1
                    active true
                    last-app-update 2023-01-26T14:59:31.385Z
                    next-hop-group 538392877120
                    next-hop-group-network-instance IP-VRF-1
```

```
                    resilient-hash false
                    fib-programming {
                        suppressed false
                        last-successful-operation-type modify
                        last-successful-operation-timestamp 2023-01-26T14:59:31.387Z
                        pending-operation-type none
                        last-failed-operation-type none
                    }
                }
                route 20.20.20.0/24 id 5 route-type local route-owner net_inst_mgr origin-
network-instance IP-VRF-2 {
                    leakable false
                    metric 0
                    preference 0
                    active true
                    last-app-update 2023-01-26T14:59:31.384Z
                    next-hop-group 538392877082
                    next-hop-group-network-instance IP-VRF-2
                    resilient-hash false
                    fib-programming {
                        suppressed false
                        last-successful-operation-type modify
                        last-successful-operation-timestamp 2023-01-26T14:59:31.387Z
                        pending-operation-type none
                        last-failed-operation-type none
                    }
                }
                route 100.100.100.0/24 id 0 route-type bgp-evpn route-owner bgp_evpn_mgr
 origin-network-instance IP-VRF-GLOBAL {
                    leakable true
                    metric 0
                    preference 170
                    active true
                    last-app-update 2023-01-26T14:59:31.386Z
                    next-hop-group 538392877123
                    next-hop-group-network-instance IP-VRF-GLOBAL
                    resilient-hash false
                    fib-programming {
                        suppressed false
                        last-successful-operation-type modify
                        last-successful-operation-timestamp 2023-01-26T14:59:31.387Z
                        pending-operation-type none
                        last-failed-operation-type none
                    }
                }
                route 200.200.200.0/24 id 0 route-type bgp-evpn route-owner bgp_evpn_mgr
 origin-network-instance IP-VRF-GLOBAL {
                    leakable true
                    metric 0
                    preference 170
                    active true
                    last-app-update 2023-01-26T14:59:31.386Z
                    next-hop-group 538392877123
                    next-hop-group-network-instance IP-VRF-GLOBAL
                    resilient-hash false
                    fib-programming {
                        suppressed false
                        last-successful-operation-type modify
                        last-successful-operation-timestamp 2023-01-26T14:59:31.387Z
                        pending-operation-type none
                        last-failed-operation-type none
                    }
                }
                statistics {
```

```
                    active-routes 5
                    active-routes-with-ecmp 0
                    resilient-hash-routes 0
                    fib-failed-routes 0
                    total-routes 5
                }
                route-summary {
                    route-type arp-nd {
                        active-routes 1
                    }
                    route-type bgp-evpn {
                        active-routes 2
                    }
                    route-type local {
                        active-routes 2
                    }
                }
            }
            ipv6-unicast {
            }
            next-hop-group 538392877123 {
                backup-next-hop-group 0
                fib-programming {
                    last-successful-operation-type add
                    last-successful-operation-timestamp 2023-01-26T14:59:31.393Z
                    pending-operation-type none
                    last-failed-operation-type none
                }
                next-hop 0 {
                    next-hop 538392877114
                    resolved true
                }
            }
            next-hop 538392877114 {
                type indirect
                ip-address 10.0.0.4
                resolving-tunnel {
                    ip-prefix 10.0.0.4/32
                    tunnel-type vxlan
                    tunnel-owner vxlan_mgr
                }
                vxlan {
                    vni 1000
                    source-mac 1A:8B:08:FF:00:00
                    destination-mac 00:00:DC:00:00:01
                }
            }
        }
    }
```

9.10 Configuring interface references

An interface reference is a user-defined name for a configured interface and subinterface within a network-instance. You can refer to the interface by the defined name within the context of the network instance.

The following example configures interface ethernet-1/10, subinterface 1 on the SR Linux device. In network instance black, an interface named red is created. Interface red is defined as interface ethernet-1/10, subinterface 1.

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

```
# info with-context interface
    interface ethernet-1/10 {
        subinterface 1 {
        }
    }
```

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance black
    network-instance black {
        interface red {
            interface-ref {
                interface ethernet-1/10
                subinterface 1
            }
        }
    }
```

In the context of network instance black, you can use the name red to refer to interface ethernet-1/10, subinterface 1. For example:

```
--{ + running }--[ network-instance black ]--
# show interfaces red
============================================================================
Net instance    : black
Interface       : ethernet-1/10.1 (red)
Type            : routed
Oper state      : down
Oper down reason: subif-down
Ip mtu          : 1500
============================================================================
```

10 SR Linux applications

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 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, SR Linux supports installation of user-defined applications, which are managed and configured in the same way as the default SR Linux applications. User-defined SR Linux applications can be loaded, reloaded, and unloaded from the system as necessary.

10.1 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

```
--{ candidate }--[  ]--
# bash
# 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/.
# exit
```

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 |
  +-----------+-----+---------+----------------------+--------------------------+
```

10.2 Starting an application

Procedure

To start an SR Linux application instance, use the **start** option in the **tools system app-management** command.

Example

To start an SR Linux application instance:

```
--{ running }--[  ]--
# tools system app-management application mpls_mgr start
/system/app-management/application[name=mpls_mgr]:
    Application 'mpls_mgr' was started
```

10.3 Restarting applications

SR Linux applications that are currently running can be restarted; that is, stopped then started again. SR Linux supports the following types of application restarts: warm restart, cold restart, and hot restart. The difference between these restart types lies in how the application state is maintained in the IDB:

- **warm restart**

  Warm restart allows forwarding to continue based on the previous state of the application. A warm restart causes the application to leave its state information in IDB during the restart, then recover it after

the restart. This restart type results in less disruption to surrounding applications that depend on the restarting application's state.

- **cold restart**

  Cold restart results in a typical stop and start of the application, including purging its state in IDB.

- **hot restart**

  When an application crashes on a system with redundancy configured, the application attempts hot restart. Triggering a hot restart initiates a switchover. Hot restart enables forwarding and preserves the application's state information in IDB and allows for the rapid restart of applications in case of a failure, minimizing downtime and ensuring continuous operation. For more information, see Non-Stop Routing.

You can determine what type of restart is supported by each application by executing the **info from state** command; for example:

```
# info from state system app-management application chassis_mgr supported-restart-types
    system {
        app-management {
            application chassis_mgr {
                supported-restart-types [
                    cold
                    warm
                ]
            }
        }
    }
```

Not all SR Linux applications support warm restart. For a list of applications that support warm restart, see the *SR Linux Software Release Notes*.

Applications can be restarted automatically by `app_mgr` (for example, if an application fails and needs to be restarted) or you can restart an application manually from the CLI using a **tools** command. When `app_mgr` restarts an application, the application is warm-restarted if the application supports it; otherwise the application is cold-restarted. If you restart an application using a **tools** command, you can specify whether the application is cold or warm restarted. If you do not specify the restart type, the application is warm-restarted if the application supports it; otherwise the application is cold-restarted.

10.3.1 Restarting an application

Procedure

You can restart an application from the SR Linux CLI. The application can be cold-restarted, warm-restarted (if the application supports warm restart). Using the **info from state** command, you can display the type of restart (cold or warm) that was used the last time the application was restarted.

Example: Restart an SR Linux application instance

```
--{ running }--[  ]--
# tools system app-management application chassis_mgr restart cold
/system/app-management/application[name=chassis_mgr]:
    Application 'chassis_mgr' was killed with signal 9

/system/app-management/application[name=chassis_mgr]:
    Application 'chassis_mgr' was restarted
```

Example: Display the type of restart used the last time the application was restarted

```
--{ running }--[  ]--
# info with-context from state system app-management application chassis_mgr last-start-
type
    system {
        app-management {
            application chassis_mgr {
                last-start-type cold
            }
        }
    }
```

10.4 Terminating an application

Procedure

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/core` directory.

- **kill**

  Terminates the application immediately, without allowing it to clean up before exiting.

Example: Terminate an application gracefully

```
--{ candidate }--[  ]--
# tools system app-management application mpls_mgr stop
/system/app-management/application[name=mpls_mgr]:
    Application 'mpls_mgr' was killed with signal 15
```

Example: Terminate an application and generate a core dump

```
--{ candidate }--[  ]--
# tools system app-management application mpls_mgr quit
/system/app-management/application[name=mpls_mgr]:
    Application 'mpls_mgr' was killed with signal 3
```

Example: Terminate an application immediately

```
--{ candidate }--[  ]--
# tools system app-management application mpls_mgr kill
/system/app-management/application[name=mpls_mgr]:
    Application 'mpls_mgr' was killed with signal 9
```

10.5 Reloading application configuration

Procedure

To reload the application configuration, reload the application manager. Reloading the application manager causes it to process any changes in the installed applications' YAML configuration and restart the applications. Applications that are no longer present in the system are stopped, and their YANG modules are removed from the management server. Applications removed from the system have their nodes or augmentations removed from the system schema.

Example

```
--{ * running }--[  ]--
# tools system app-management application app_mgr reload
```

10.6 Clearing application statistics

Procedure

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:

```
--{ running }--[  ]--
# tools system app-management application mpls_mgr statistics clear
```

10.7 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 16: Restricted operations for SR Linux applications lists the restricted operations for each SR Linux application.

Table 16: Restricted operations for SR Linux applications

Application	Restricted operations
aaa_mgr	reload
acl_mgr	reload
app_mgr	start, stop, restart, quit, kill

Application	Restricted operations
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
grpc_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
plcy_mgr	reload
qos_mgr	reload
sdk_mgr	reload
static_route_mgr	reload
supportd	reload
xdp_cpm	stop, restart, quit, kill, reload
xdp_lc	reload

Restricted options are specified in the `restricted-operations` setting in the YAML file for the application.

10.8 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 stops responding five 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

```
--{ running }--[  ]--
# show system application
+--------------+-----+--------------------+----------------------+--------------------------+
|    Name      | PID |       State        |        Version       |       Last Change        |
+==============+=====+====================+======================+==========================+
| aaa_mgr      | 242 | error              | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.967Z |
| acl_mgr      | 193 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.967Z |
| app_mgr      | 118 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:11.161Z |
| arp_nd_mgr   | 104 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.967Z |
| bfd_mgr      |     | waiting-for-config |                       |                          |
| bgp_mgr      | 109 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:13.156Z |
| chassis_mgr  | 115 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.967Z |
| dev_mgr      | 150 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.001Z |
| fib_mgr      | 168 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.968Z |
| grpc_server  | 157 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:13.296Z |
| idb_server   | 171 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.228Z |
| igmp_mgr     |     | waiting-for-config |                       |                          |
| isis_mgr     |     | waiting-for-config |                       |                          |
| json_rpc     | 166 | running            |                       | 2022-01-21T20:15:13.234Z |
| label_mgr    | 139 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:13.186Z |
| lag_mgr      | 103 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.968Z |
| ldp_mgr      |     | waiting-for-config |                       |                          |
| linux_mgr    | 116 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.968Z |
```

```
| lldp_mgr     | 149 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:13.206Z |
| log_mgr      | 127 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.968Z |
| mgmt_server  | 149 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.968Z |
| mirror_mgr   |     | waiting-for-config |                       |                          |
| mpls_mgr     |     | waiting-for-config |                       |                          |
| net_inst_mgr | 160 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.968Z |
| oam_mgr      | 160 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:13.231Z |
| ospf_mgr     |     | waiting-for-config |                       |                          |
| p4rt_server  | 641 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T21:56:47.935Z |
| plcy_mgr     | 177 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:13.242Z |
| qos_mgr      | 186 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:13.322Z |
| sshd-mgmt    | 192 | running            |                       | 2022-01-21T20:15:19.710Z |
| xdp_cpm      | 197 | running            | v22.3.0-34-ge0ee326f8 | 2022-01-21T20:15:10.968Z |
| xdp_lc_1     | 108 | running            |                       | 2022-01-21T20:15:10.968Z |
+--------------+-----+--------------------+-----------------------+--------------------------+
```

Step 2. Use the **info from state** command to check the current failure action settings for the application to configure.

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 action 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

These settings are highlighted in the following example:

Example

```
--{ running }--[  ]--
# info with-context from state system app-management application aaa_mgr
    system {
        app-management {
            application aaa_mgr {
                pid 242
                state error
                last-change 2022-01-21T20:15:10.967Z
                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 v22.3.0-34-ge0ee326f8
                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
                    ]
                }
            }
        }
    }
```

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 theSR 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 five 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

```
--{ running }--[  ]--
# 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

```
--{ running }--[  ]--
# info with-context from state system app-management application aaa_mgr
    system {
        app-management {
            application aaa_mgr {
                pid 242
                state running
                last-change 2022-01-21T20:15:10.967Z
                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 v22.3.0-34-ge0ee326f8
                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
                    ]
                }
            }
        }
    }
```

10.9 Removing an application from the system

About this task

To remove an application from the system, remove the application files from the SR Linux directories where they are located, then reload the application manager.

When an application is removed from the system, SR Linux stops sending updates for the paths that the application would populate. If the application is subsequently reloaded, SR Linux resumes sending updates.

For active CLI sessions, the schema is updated to remove the application. If an active CLI session exists in a context that is no longer present because of the application being removed, the CLI context is changed to the next highest valid context.

User-defined applications can be removed from the system, but removing Nokia-provided SR Linux applications is not supported.

In the following example, the `fib_agent` application, consisting of files named `fib_agent.yml`, `fib_agent.sh`, `fib_agent.py`, and `fib_agent.yang`, is removed from the system.

Procedure

Step 1. Remove the application files from the SR Linux directories.

Example

```
--{ candidate }--[  ]--
# bash
# rm /etc/opt/srlinux/appmgr/fib_agent.yml
# rm /user_agents/fib_agent.sh
# rm /user_agents/fib_agent.py
# rm /yang/fib_agent.yang
# exit
```

Step 2. From the SR Linux CLI, reload the application manager.

Example

```
--{ candidate }--[ ]--
# tools system app-management application app_mgr reload
```

The application manager stops any application that is no longer present and removes the application's YANG module from the management server.

10.10 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.

10.10.1 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.

10.10.1.1 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 following shows the default `cgroup_profile.json` file definition:

```
{
 "profiles": [
  {
   "name": "workload.slice",
   "path": "workload.slice",
   "controller": {
    "memory": {
     "max": 0.8,
     "swap_max": 0,
     "low": 0,
     "max_unoccupied_mem": "1024"
    },
    "cpu": {
     "weight": "1000",
     "period": "100000",
     "quota": "0"
    },
    "cpuset": {
     "cpus": "all",
     "mems": ""
    }
   }
  },
  {
   "name": "workload.primary",
   "path": "workload.slice/primary",
   "controller": {
    "memory": {
     "max": 0.8,
     "swap_max": 0,
     "low": 0,
     "max_unoccupied_mem": "1024"
    },
    "cpu": {
     "weight": "100",
     "period": "100000",
     "quota": "0"
    },
    "cpuset": {
     "cpus": "all",
     "mems": ""
    }
   }
  },
  {
   "name": "workload.primary.nondatapath",
   "path": "workload.slice/primary/nondatapath",
   "controller": {
```

```
   "cpu": {
    "weight": "100",
    "period": "100000",
    "quota": "0"
   },
   "cpuset": {
    "cpus": "",
    "mems": ""
   }
  }
 },
 {
  "name": "workload.primary.datapath",
  "path": "workload.slice/primary/datapath",
  "controller": {
   "cpu": {
    "weight": "100",
    "period": "100000",
    "quota": "0"
   },
   "cpuset": {
    "cpus": "",
    "mems": ""
   }
  }
 },
 {
  "name": "workload.primary.devicemgr",
  "path": "workload.slice/primary/devicemgr",
  "controller": {
   "cpu": {
    "weight": "100",
    "period": "100000",
    "quota": "0"
   },
   "cpuset": {
    "cpus": "",
    "mems": ""
   }
  }
 },
 {
  "name": "workload.primary.appmgr",
  "path": "workload.slice/primary/appmgr",
  "controller": {
   "cpu": {
    "weight": "1000",
    "period": "100000",
    "quota": "0"
   },
   "cpuset": {
    "cpus": "",
    "mems": ""
   }
  }
 },
 {
  "name": "workload.secondary",
  "path": "workload.slice/secondary",
  "controller": {
   "memory": {
    "max": 0.5,
    "swap_max": 0,
    "low": 0,
```

```
     "max_unoccupied_mem": "0"
    },
    "cpu": {
     "weight": "100",
     "period": "100000",
     "quota": "0"
    },
    "cpuset": {
     "cpus": "",
     "mems": ""
    }
   }
  },
  {
   "name": "workload.secondary.default",
   "path": "workload.slice/secondary/default",
   "controller": {
    "memory": {
     "max": 0.5,
     "swap_max": 0,
     "low": 0,
     "max_unoccupied_mem": "0"
    },
    "cpu": {
     "weight": "100",
     "period": "100000",
     "quota": "0"
    },
    "cpuset": {
     "cpus": "",
     "mems": ""
    }
   }
  },
  {
   "name": "workload.secondary.eventmgr",
   "path": "workload.slice/secondary/eventmgr",
   "controller": {
    "memory": {
     "max": 0.1,
     "swap_max": 0,
     "low": 0,
     "max_unoccupied_mem": "0"
    },
    "cpu": {
     "weight": "100",
     "period": "100000",
     "quota": "0"
    },
    "cpuset": {
     "cpus": "",
     "mems": ""
    }
   }
  },
  {
   "name": "userload.default",
   "path": "userload.slice/default",
   "controller": {
    "memory": {
     "max": 0.25,
     "swap_max": 0,
     "low": 0,
     "max_unoccupied_mem": "0"
```

```
    },
    "cpu": {
     "weight": "100",
     "period": "100000",
     "quota": "0"
    },
    "cpuset": {
     "cpus": "",
     "mems": ""
    }
   }
  }
 ]
}
```

Table 17: Default cgroup profile parameters describes the default cgroup profile parameters.

Table 17: Default cgroup profile parameters

Parameter	Description
name	The cgroup profile name. Type: string
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.memory	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.
controller.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 must 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**

Parameter	Description
	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.
controller.cpuset	CPU usage information for the cgroup: • **cpus** This value indicates the CPUs used by the cgroup. This value 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.

10.10.1.2 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 the following Nokia default behaviors:

- 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.

10.10.1.3 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.

10.10.1.3.1 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 following procedure describes the steps and the outputs.

The preceding configuration 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 10 000 (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 following output 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 following output 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"
    }
   }
  }
 ]
}
```

10.10.2 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. 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 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.

10.10.2.1 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 18: OOM adjust score per process lists the OOM adjust score for each process.

Table 18: OOM adjust score per process

Process name	OOM adjust score
aaa_mgr	0
acl_mgr	0
app_mgr	–998

Process name	OOM adjust score
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
grpc_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
mcid_mgr	0
mgmt_server	–200
mpls_mgr	0
net_inst_mgr	–200
oam_mgr	500
ospf_mgr	0

Process name	OOM adjust score
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

10.10.3 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 OOM 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 following output.

```
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"
```

10.10.4 Debugging cgroups

Cgroup debugging capability is available through:

- SR Linux CLI commands
- Linux-provided CLI commands

10.10.4.1 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

10.10.4.1.1 Checking existing cgroup usage

The following output is an example of checking existing cgroup usage.

```
--{ running }--[  ]--
# info from state platform control A cgroup *
    cgroup /mnt/cgroup/unified/userload.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
            memory-events {
                low 0
                high 0
                max 0
                oom 0
                oom-kill 0
            }
```

```
        }
        cpuacct-statistics {
            user 0
            system 0
        }
    cgroup /mnt/cgroup/unified/workload.slice {
        memory-statistics {
            current 5887053824
            current-swap 0
            anon 5344722944
            kernel-stack 3178496
            slab 21182248
            sock 0
            anon-thp 2734686208
            file 249106432
            file-writeback 0
            file-dirty 0
            memory-events {
                low 0
                high 0
                max 0
                oom 0
                oom-kill 0
            }
        }
        cpuacct-statistics {
            user 180449217810
            system 31712573916
        }
    }
    cgroup /mnt/cgroup/unified/workload.slice/primary {
        memory-statistics {
            current 5840982016
            current-swap 0
            anon 5342904320
            kernel-stack 3162112
            slab 21072224
            sock 0
            anon-thp 2734686208
            file 205103104
            file-writeback 0
            file-dirty 0
            memory-events {
                low 0
                high 0
                max 0
                oom 0
                oom-kill 0
            }
        }
        cpuacct-statistics {
            user 180448931554
            system 31712492691
        }
    }
|
|
|
    cgroup /mnt/cgroup/unified/workload.slice/secondary {
        memory-statistics {
            current 2015232
            current-swap 0
            anon 1818624
            kernel-stack 16384
```

```
            slab 75232
            sock 0
            anon-thp 0
            file 4096
            file-writeback 0
            file-dirty 0
            memory-events {
                low 0
                high 0
                max 0
                oom 0
                oom-kill 0
            }
        }
        cpuacct-statistics {
            user 291582
            system 79431
        }
    }
    cgroup /mnt/cgroup/unified/workload.slice/secondary/default {
        memory-statistics {
            current 1994752
            current-swap 0
            anon 1818624
            kernel-stack 16384
            slab 75232
            sock 0
            anon-thp 0
            file 4096
            file-writeback 0
            file-dirty 0
            memory-events {
                low 0
                high 0
                max 0
                oom 0
                oom-kill 0
            }
        }
        cpuacct-statistics {
            user 291582
            system 79431
        }
    }
    cgroup /mnt/cgroup/unified/workload.slice/secondary/eventmgr {
        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
            memory-events {
                low 0
                high 0
                max 0
                oom 0
                oom-kill 0
            }
        }
```

```
        cpuacct-statistics {
            user 0
            system 0
        }
    }
```

10.10.4.1.2 Showing current OOM adjust scores

The following output is an example of showing information about the current OOM adjust scores for all applications that are managed by the app_mgr.

```
--{ running }--[  ]--
# info with-context 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
            }
        }
    }
```

10.10.4.1.3 Showing cgroup information

The following output 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
    application aaa_mgr {
        cgroup /mnt/cgroup/unified/workload.slice/primary/nondatapath
    }
    application acl_mgr {
        cgroup /mnt/cgroup/unified/workload.slice/primary/nondatapath
    }
    application arp_nd_mgr {
        cgroup /mnt/cgroup/unified/workload.slice/primary/nondatapath
    }
    application bfd_mgr {
        cgroup /mnt/cgroup/unified/workload.slice/primary/nondatapath
    }
```

```
    application chassis_mgr {
        cgroup /mnt/cgroup/unified/workload.slice/primary/nondatapath
    }
    application dev_mgr {
        cgroup /mnt/cgroup/unified/workload.slice/primary/devicemgr
    }
    application dhcp_client_mgr {
        cgroup /mnt/cgroup/unified/workload.slice/primary/nondatapath
    }
    application evpn_mgr {
        cgroup /mnt/cgroup/unified/workload.slice/primary/nondatapath
    }
    application fib_mgr {
        cgroup /mnt/cgroup/unified/workload.slice/primary/nondatapath
    }
    application grpc_server {
        cgroup /mnt/cgroup/unified/workload.slice/primary/nondatapath
|
|
|
    application supportd {
        cgroup /mnt/cgroup/unified/workload.slice/primary/nondatapath
    }
    application vxlan_mgr {
        cgroup /mnt/cgroup/unified/workload.slice/primary/nondatapath
    }
    application xdp_lc_1 {
        cgroup /mnt/cgroup/unified/workload.slice/primary/datapath
    }
```

10.10.4.1.4 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 following output shows an example.

```
--{ running }--[  ]--
# tools system cgroup command pgrep cgroup workload.slice/primary/nondatapath
|  Pid  |       Process       |
+=======+=====================+
| 22368 | sr_supportd         |
| 22481 | top                 |
| 23140 | sr_idb_server       |
| 23284 | sr_aaa_mgr          |
| 23350 | sr_acl_mgr          |
| 23401 | sr_arp_nd_mgr       |
| 23465 | sr_bfd_mgr          |
| 23516 | sr_chassis_mgr      |
| 23563 | sr_dhcp_client_mgr  |
| 23602 | sr_evpn_mgr         |
| 23671 | sr_fib_mgr          |
| 23741 | sr_l2_mac_learn_mgr |
| 23795 | sr_l2_mac_mgr       |
| 23848 | sr_lag_mgr          |
| 23891 | sr_linux_mgr        |
| 23963 | sr_log_mgr          |
| 24023 | sr_mcid_mgr         |
| 24085 | sr_mfib_mgr         |
| 24131 | sr_mgmt_server      |
| 24188 | sr_net_inst_mgr     |
| 24250 | sr_radius_mgr       |
| 24289 | sr_sflow_sample_mgr |
```

```
| 24346 | sr_stat_id_mgr      |
| 25575 | sr_grpc_server      |
| 25614 | sr_plcy_mgr         |
| 25655 | sr_qos_mgr          |
| 25693 | sr_vxlan_mgr        |
| 69524 | bash                |
| 69541 | python              |
+-------+---------------------+
```

10.10.4.2 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 following output shows an example of the **systemd-cgls** command.

```
[linuxadmin@srlinux unified]$ systemd-cgls
Control group /:
-.slice
├─workload.slice (#6685)
│ ├─primary (#6750)
│ │ ├─nondatapath (#6792)
│ │ │ ├─22368 ./sr_supportd --server-mode
│ │ │ ├─22481 top -b -d 600 -H -i -w 512
│ │ │ ├─23140 ./sr_idb_server
│ │ │ ├─23284 ./sr_aaa_mgr
│ │ │ ├─23350 ./sr_acl_mgr
│ │ │ ├─23401 ./sr_arp_nd_mgr
│ │ │ ├─23465 ./sr_bfd_mgr
│ │ │ ├─23516 ./sr_chassis_mgr
│ │ │ ├─23563 ./sr_dhcp_client_mgr
│ │ │ ├─23602 ./sr_evpn_mgr
│ │ │ ├─23671 ./sr_fib_mgr
│ │ │ ├─23741 ./sr_l2_mac_learn_mgr
│ │ │ ├─23795 ./sr_l2_mac_mgr
│ │ │ ├─23848 ./sr_lag_mgr
│ │ │ ├─23891 ./sr_linux_mgr
│ │ │ ├─23963 ./sr_log_mgr
│ │ │ ├─24023 ./sr_mcid_mgr
│ │ │ ├─24085 ./sr_mfib_mgr
│ │ │ ├─24131 ./sr_mgmt_server
│ │ │ ├─24188 ./sr_net_inst_mgr
│ │ │ ├─24250 ./sr_radius_mgr
│ │ │ ├─24289 ./sr_sflow_sample_mgr
│ │ │ ├─24346 ./sr_stat_id_mgr
│ │ │ ├─25575 ./sr_grpc_server
│ │ │ ├─25614 ./sr_plcy_mgr
│ │ │ ├─25655 ./sr_qos_mgr
│ │ │ └─25693 ./sr_vxlan_mgr
│ │ ├─devicemgr (#6846)
│ │ │ └─22567 ./sr_device_mgr
│ │ ├─datapath (#6819)
│ │ │ └─24385 sr_xdp_lc_1 --slot_num 1 --complex_num 2
│ │ └─appmgr (#6873)
│ │   └─20843 ./sr_app_mgr
│ └─secondary (#6900)
│   └─default (#6942)
```

```
|     └─26090 sshd: /usr/sbin/sshd -f /etc/ssh/sshd_config_mgmt
```

The **systemd-cgtop** command dumps the current usage of each cgroup. The following output 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      -        -        -        -
```

11 Control plane resource monitoring

SR Linux allows you to set thresholds for monitoring the following control plane resources:

- `mpls-next-hops`
- `tunnels` This refers to all tunnels (MPLS, IP-in-IP, GRE, VXLAN).
- `originating-ip-in-ip-tunnels`
- `originating-ip-gre-tunnels`
- `vp-lag-groups`
- `protect-groups`

For each of these resources, you can specify an `upper-threshold-set` value (a percentage from 0-100) and an `upper-threshold-clear` value (a percentage from 0-100); these values apply system-wide.

- The `upper-threshold-set` value sets the threshold that triggers the generation of a WARNING log and the setting of `used-upper-threshold-exceeded` to `true` whenever the utilization of the resource reaches this value in a rising direction.
- The `upper-threshold-clear` value sets the threshold that triggers the generation of a NOTICE log and the setting of `used-upper-threshold-exceeded` to `false` whenever the utilization of the resource reaches this value in a falling direction.

11.1 Configuring thresholds for control plane resources

Procedure

To configure monitoring for control plane resources, set values for the `upper-threshold-set` and `upper-threshold-clear` parameters for the resource you want to monitor.

Example: Configure thresholds for a system resource

The following example configures thresholds for a system resource.

```
--{ * candidate shared default }--[  ]--
# info with-context system utilization
    system {
        utilization {
            resource ip-gre-tunnels {
            }
            resource tunnels {
                upper-threshold-set 75
                upper-threshold-clear 60
            }
        }
    }
```

Example: Display resource thresholds and statistics

The following example displays the current thresholds and statistics for each resource.

```
--{ + running }--[  ]--
# info with-context from state system utilization resource *
    system {
        utilization {
            resource ip-gre-tunnels {
                upper-threshold-set 75
                upper-threshold-clear 60
                used-percent 0
                used-entries 0
                free-entries 2048
                used-upper-threshold-exceeded false
            }
            resource ip-in-ip-tunnels {
                upper-threshold-set 90
                upper-threshold-clear 70
                used-percent 0
                used-entries 0
                free-entries 4096
                used-upper-threshold-exceeded false
            }
            resource mpls-next-hops {
                upper-threshold-set 90
                upper-threshold-clear 70
                used-percent 0
                used-entries 0
                free-entries 65536
                used-upper-threshold-exceeded false
            }
            resource protect-groups {
                upper-threshold-set 90
                upper-threshold-clear 70
                used-percent 0
                used-entries 0
                free-entries 4096
                used-upper-threshold-exceeded false
            }
            resource tunnels {
                upper-threshold-set 90
                upper-threshold-clear 70
                used-percent 0
                used-entries 0
                free-entries 16000
                used-upper-threshold-exceeded false
            }
        }
    }
```

12 Datapath resource management

The 7220 IXR-D1, 7220 IXR-D2, 7220 IXR-D3, 7220 IXR-D4, 7220 IXR-D5, and 7220 IXR-H*x* have forwarding pipelines that access a set of shared banks that are 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 through different configuration parameters as follows:

- The 7220 IXR-D1, 7220 IXR-D2, and 7220 IXR-D3 support configuration of the **alpm** mode (ALPM = algorithmic longest prefix match), the **requested-extra-ip-host-entries** value, and the **ipv6-128bit-lpm-entries** value.
  - For alpm, three settings are supported: `disabled`, `enabled` and `high-scale`. **alpm disabled** completely disables ALPM mode, and provides maximum resources to be split between IP host entries and MAC table entries. **alpm enabled** is only available on the 7220 IXR-D1 and uses more shared table resources, suitable for supporting moderate IP FIB scale. **alpm high-scale** is only available on the 7220 IXR-D2 and 7220 IXR-D3 and uses all available shared resources for IP FIB tables, resulting in only the base allocation for IP host entries and MAC table entries.
  - **requested-extra-ip-host-entries** should be configured with the number of IP host entries needed beyond the base allocation. It is possible, depending on the alpm mode setting that not all the requested entries can be allocated. The allocated number of entries can viewed in **info from state**, as discussed in the section Displaying datapath resource management information
- The 7220 IXR-D4 and 7220 IXR-D5 support configuration of only the **alpm** mode, and only two settings are available:
  - **enabled** – enables ALPM using less shared table resources, supporting only moderate IP FIB scale
  - **high-scale** – enables ALPM using more shared table resources, supporting the highest possible IP FIB scale
- The 7220 IXR-H2 and 7220 IXR-H3 support the configuration of the **ipv6-128bit-lpm-entries** value. For more information, see LPM table partitioning.

12.1 LPM table partitioning

On the 7220 IXR-D1, 7220 IXR-D2, 7220 IXR-D3, 7220 IXR-H2, and 7220 IXR-H3, IP FIB scale depends on the partitioning of the hardware LPM table.

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.

12.2 Changing datapath resource management settings

Procedure

You can change the datapath resource management settings discussed in LPM table partitioning in order to provider higher scale for one entry type at the expense of lower scale for another entry type.

This is illustrated in the following examples.

Example: Configure Settings for the 7220 IXR-D1

The following example changes datapath resource management settings on the 7220 IXR-D1. This configuration enables Algorithmic Longest Prefix Matching (ALPM) to achieve higher IP FIB scale and it requests an extra 32K IPv4 host entries on top of the fixed base amount of 16K entries. Any remaining datapath resources, if any, will be allocated to storing MAC addresses. This configuration also sets the hardware LPM table partition to the default value of 1024.

```
--{ * candidate shared default }--[  ]--
# info with-context 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
            }
        }
    }
```

Example: Configure Settings for the 7220 IXR-D5

The following example changes datapath resource management settings on the 7220 IXR-D5. This configuration enables ALPM (but not in the **high-scale** mode) to achieve moderate IP FIB scale and leave more resources for IP host and MAC table entries.

```
--{ * candidate shared default }--[  ]--
# info with-context platform resource-management unified-forwarding-resources
    platform {
        resource-management {
            unified-forwarding-resources {
                alpm enabled
            }
        }
    }
```

Example: Restart XDP to activate changes

Datapath resource management configuration changes do not take effect immediately when the changes are committed; they take effect the next time XDP is restarted. The following example shows a tools command that restarts XDP:

```
--{ running }--[  ]--
# tools system app-management application xdp_lc_1 restart
/system/app-management/application[name=xdp_lc_1]:
    Application 'xdp_lc_1' was killed with signal 9

/system/app-management/application[name=xdp_lc_1]:
    Application 'xdp_lc_1' was restarted
```

On the 7220 IXR-D1, 7220 IXR-D2, 7220 IXR-D3, 7220 IXR-D4, 7220 IXR-D5, and 7220 IXR-H*x*, this command also restarts XDP-CPM, which causes a temporary loss of management connectivity.

12.3 Displaying datapath resource management information

Procedure

Use the **info from state** command to display the allocated number of extra host entries and extra MAC address entries allocated from datapath resource management shared banks.

Example

```
--{ running }--[  ]--
# info with-context 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.

13 Datapath resource monitoring

On 7215 IXS, 7220 IXR, 7250 IXR, and 7730 SXR devices, SR Linux exposes utilization information about the following datapath resources, which are limited in number and potentially can be exhausted:

- ACL-related resources
- TCAM-related resources
- MTU-related resources (only applicable to 7250 IXR devices)
- QoS profile-related resources
- Buffer memory
- XDP forwarding-related resources
- ASIC-specific forwarding-related resources

For each of these resources, you can specify a **rising-threshold-log** value (in percentage, 0 to 100) and a **falling-threshold-log** value (in percentage, 0 to 100). These values apply system-wide.

The **rising-threshold-log** value sets the threshold that triggers the generation of the `systemResourceHighUtilization` log message with `warning` severity whenever the utilization of the datapath resource in any line card (if applicable), forwarding-complex, or pipeline (if applicable) reaches this value in a rising direction. The log message captures the following information:

- Line card slot number (if applicable)
- Forwarding-complex name, for example “0”
- Pipeline number (if applicable)
- Resource name, for example `ip-lpm-routes`

The **falling-threshold-log** value sets the threshold that triggers the generation of the `systemResourceHighUtilizationLowered` log message with `notice` severity whenever the utilization of the datapath resource reaches this value in a falling direction. This log message captures the same information as the **rising-threshold-log** message.

13.1 Configuring thresholds for datapath resources

Procedure

To configure monitoring for datapath resources, use the **platform resource-monitoring** command to set values for the **rising-threshold-log** and **falling-threshold-log** parameters for the resource you want to monitor.

Example: Set thresholds for resource usage by input IPv4 filter instances

The following example sets thresholds for resource usage by input IPv4 filter instances.

```
--{ * candidate shared default }--[  ]--
# info with-context platform resource-monitoring
    platform {
        resource-monitoring {
```

```
            acl {
                resource input-ipv4-filter-instances {
                    rising-threshold-log 90
                    falling-threshold-log 75
                }
            }
        }
    }
```

Example: Set thresholds for TCAM resource usage by IPv4 CPM filters

The following example sets thresholds for TCAM resource usage by IPv4 CPM filters.

```
--{ * candidate shared default }--[  ]--
# info with-context platform resource-monitoring
    platform {
        resource-monitoring {
            tcam {
                resource cpm-capture-ipv4 {
                    rising-threshold-log 90
                    falling-threshold-log 75
                }
            }
        }
    }
```

14 Flexible counter bank allocation

Note: This feature is supported on 7250 IXR Gen 2c+ and 7250 IXR Gen 3 platforms.

SR Linux devices allow flexible counter bank allocation for the following applications:

7250 IXR Gen 2c+

- Early Congestion Notification (ECN) statistics
- LSP statistics (ingress and egress)
- IP-in-IP tunnel statistics

7250 IXR Gen 3

- Early Congestion Notification (ECN) statistics
- IP-in-IP tunnel statistics

Some counter banks are pre-assigned for essential applications, for example, CPM filter counters, interface queue statistics, and so on. These pre-assigned banks cannot be modified. Other banks are free and available for assignment to the applications listed above. The number of banks you assign, along with their size, determines how many statistics can be collected. To view the list of banks you can allocate to an application, use the **info from state with-context platform linecard forwarding-complex counter-banks** command. See Displaying bank allocations on line cards for more information.

For each SR Linux device, you can select one of the supported applications and assign one or more free counter banks identified by unique IDs. Banks come in different sizes depending on the SR Linux device. You must match the application's scale requirement to a bank large enough to store all counters. If a single bank is not large enough, you can allocate multiple banks to expand capacity. For example, assigning **bank-id** *1* to **ingress-lsp-statistics**, stores LSP ingress statistics in bank 1 on a specific device. Adding **bank-id** *4* to the same application expands the allocation, therefore increasing the capacity. See Performing counter bank allocations for more inforation. In addition to bank allocation, the applications may also have specific bank size or scaling requirements.

Counter banks available on 7250 IXR Gen 2c+

The following table lists the free counter banks with IDs and sizes available for allocation when a 7250 IXR Gen 2c+ device boots.

Table 19: Free counter banks available on 7250 IXR Gen 2c+

Bank ID	Bank size
1, 4	4K
13, 15	8K
19, 28, 29, 30, 31, 32, 33, 34	16K

Counter banks available on 7250 IXR Gen 3

The following table lists the free counter bank with IDs and sizes available for allocation when a 7250 IXR Gen 3 device boots.

Table 20: Free counter banks available for allocation on 7250 IXR Gen 3

Bank ID	Bank size
1, 2, 4, 11, 12	6K
21, 22, 23, 24	12K
30, 31	24K

Reboot requirements

Reboot requirements for flexible counter bank allocation are as follows:

- Reboot is not required for the following tasks:
  - Allocating a free bank to an application.
  - Adding a free bank to an existing application.
- Reboot is required for the following tasks:
  - An application evacuating a bank.
  - Replacing a bank with another bank, smaller or larger

The show command, **info from state with-context platform resource-management counter-banks** displays the pending allocations that require system reboot. See Verifying reboot status of bank allocations for more information.

Bank allocation considerations

The following considerations apply while allocating counter banks:

- If no bank is allocated or allocation fails and the application enables statistics, the node returns statistics as all zeros. The node reports the allocation failure back to the application. The node does not manage retries or maintain retry queues. The application retries using a bank when it becomes available.
- If counter banks are not available for additional entries, the node reports the shortage.
- Applications define their own maximum limits. Even if banks are available, statistics collection stops after the application reaches the maximum limit.

14.1 Performing counter bank allocations

Procedure

To allocate a bank to an application, use the **platform resource-management counter-banks forwarding-complex-type application bank-allocation** command, specifying the SR Linux device type, application, and the bank ID as shown in the example.

Example: Allocating a bank to 7250 IXR Gen 2c+ egress LSP statistics

This example allocates bank 1 to 7250 IXR Gen 2c+ egress LSP statistics.

```
--{ + candidate shared default }--[  ]--
# info with-context platform resource-management counter-banks
    platform {
        resource-management {
            counter-banks {
                forwarding-complex-type Gen2c+ {
                    application egress-lsp-stats {
                        bank-allocation 1 {
                        }
                    }
                }
            }
        }
    }
```

14.2 Displaying bank allocations on line cards

Procedure

To view the bank allocation on line cards, use the **info from state with-context platform linecard forwarding-complex counter-banks** command specifying the line card and forwarding complex.

Example: View bank allocation on a line card

This example displays the bank allocation on line card 1.

```
--{ + candidate shared default }--[  ]--
# info from state with-context platform linecard 1 forwarding-complex 1 counter-banks
    platform {
        linecard 1 {
            forwarding-complex 1 {
                counter-banks {
                    bank 1 {
                        bank-size 4K
                        application ingress-lsp-stats {
                            entries 4096
                        }
                    }
                    bank 4 {
                        bank-size 4K
                        application ingress-lsp-stats {
                            entries 4096
                        }
                    }
                    bank 13 {
                        bank-size 8K
                        application unallocated {
                        }
                    }
                    bank 15 {
                        bank-size 8K
                        application unallocated {
                        }
                    }
                    bank 19 {
```

```
                    bank-size 16K
                    application unallocated {
                    }
                }
                bank 28 {
                    bank-size 16K
                    application unallocated {
                    }
                }
                bank 29 {
                    bank-size 16K
                    application unallocated {
                    }
                }
                bank 30 {
                    bank-size 16K
                    application unallocated {
                    }
                }
                bank 31 {
                    bank-size 16K
                    application unallocated {
                    }
                }
                bank 32 {
                    bank-size 16K
                    application unallocated {
                    }
                }
                bank 33 {
                    bank-size 16K
                    application unallocated {
                    }
                }
                bank 34 {
                    bank-size 16K
                    application unallocated {
                    }
                }
            }
        }
    }
}
```

14.3 Verifying reboot status of bank allocations

Procedure

To verify whether a bank allocation takes effect immediately or requires a reboot, use the **info from state with-context platform resource-management counter-banks** command. Check the **chassis-reboot-required** flag in the output.

If the flag is set to **false**, the allocation is active immediately. See Reboot not required.

If the flag is set to **true**, the allocation only applies after a reboot. See Reboot required.

Example: Reboot not required

The following example shows that the bank allocation is active and reboot is not required.

```
--{ + candidate shared default }--[  ]--
# info from state with-context platform resource-management counter-banks
    platform {
        resource-management {
            counter-banks {
                chassis-reboot-required false
                forwarding-complex-type Gen2c+ {
                    application ingress-lsp-stats {
                        bank-allocation 1 {
                        }
                        bank-allocation 4 {
                        }
                    }
                }
            }
        }
    }
```

Example: Reboot required

The following example shows that the bank allocation is modified and reboot is required.

```
--{ + candidate shared default }--[  ]--
# info from state with-context platform resource-management counter-banks
    platform {
        resource-management {
            counter-banks {
                chassis-reboot-required true
                forwarding-complex-type Gen2c+ {
                    application ingress-lsp-stats {
                    }
                    application ip-in-ip-tunnel-stats {
                        bank-allocation 1 {
                        }
                        bank-allocation 4 {
                        }
                    }
                }
            }
        }
    }
```

15 Rate Limiting for ICMP messages

SR Linux limits the rate of ICMP messages that it generates and sends towards IPv4 and IPv6 hosts. It maintains a list of the 1000 most-recent source IP addresses that triggered the need to send an ICMP message to that endpoint.SR Linux does not allow an ICMP message to be transmitted to a source IP entry in the list if the last message sent to that host (of any type and code) was transmitted less than 1 second ago; as a result, there is a limit of 1000 ICMP error/redirect messages per second towards all sources.

To do this, SR Linux maintains a token bucket for each of the last 1000 IPv4 and 1000 IPv6 senders that generated traffic requiring ICMP messages to be sent back to them. The token bucket for each sender has a maximum depth of 10 packets, and can be filled or drained at a rate of 10 packets per second. You can adjust the maximum depth of the token bucket and the rate at which it is filled or drained.

15.1 Configuring rate limiting for ICMP messages

Procedure

Each token bucket has a maximum depth, counted in terms of ICMP messages, which is controlled by the **max-burst** parameter (default 10 packets), and a fill/drain rate, which is controlled by the **peak-rate** parameter (default 10 packets per second).

To configure rate limiting for ICMP messages, you can set values for the **max-burst** and **peak-rate** parameters. The token bucket state for each host is refilled at the `peak-rate` up to a maximum depth of `max-burst`.

Example: Configure rate limiting for IPv4 ICMP messages

The following example sets values for the **max-burst** and **peak-rate** parameters for IPv4 ICMP messages.

```
--{ + candidate shared default }--[  ]--
# info with-context system datapath icmp
    system {
        datapath {
            icmp {
                rate-limit-per-host {
                    peak-rate 20
                    max-burst 40
                }
            }
        }
    }
```

Example: Configure rate limiting for IPv6 ICMP messages

The following example sets values for the **max-burst** and **peak-rate** parameters for IPv6 ICMP messages.

```
--{ + candidate shared default }--[  ]--
# info with-context system datapath icmp
```

```
system {
    datapath {
        icmp6 {
            rate-limit-per-host {
                peak-rate 20
                max-burst 40
            }
        }
    }
}
```

16 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 neighbor 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.

16.1 Configuring a maintenance group

Procedure

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 with-context 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
                            ]
                        }
                    }
                }
            }
        }
    }
```

16.2 Configuring a maintenance profile

Procedure

In a maintenance profile, you specify policy changes that are applied to members of a maintenance group when maintenance mode is enabled for the group.

Example

The following example configures a routing policy and a maintenance profile that references the routing policy.

The following defines the routing policy drain-with-as-path-prepend:

```
--{ candidate shared default }--[  ]--
# info with-context 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 defines a maintenance profile that references the routing policy drain-with-as-path-prepend.

```
--{ candidate shared default }--[  ]--
# info with-context system maintenance
    system {
```

```
        maintenance {
            profile mprof1 {
                bgp {
                    import-policy drain-with-as-path-prepend
                    export-policy drain-with-as-path-prepend
                }
            }
        }
    }
```

16.3 Placing a maintenance group into maintenance mode

Procedure

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 mprof1 are applied to the resources configured in maintenance group mgroup1.

```
--{ candidate shared default }--[  ]--
# info with-context 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.
```

16.4 Taking a maintenance group out of service

Procedure

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 with-context 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.
```

16.5 Restoring the maintenance group to service

Procedure

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 with-context 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.
```

17 MACsec

Note: This feature is supported on the 7250 IXR-6e and 7250 IXR-10e (including mixed system), 7250 IXR-X1b, 7250 IXR-X3b, and 7250 IXR-X4.

Media Access Control Security (MACsec) is an industry standard security technology that provides secure communication for almost all types of traffic on Ethernet links. MACsec provides point-to-point security on Ethernet links between directly connected nodes and is capable of identifying and preventing most security threats including:

- denial of service
- intrusion
- man-in-the-middle
- masquerading
- passive wiretapping
- playback attacks

MACsec is standardized in IEEE 802.1AE.

The following sections describe the features and functions of MACsec on SR Linux.

17.1 MACsec terminology

MACsec uses Security Channels (SC) and Secure Association Keys (SAKs) to encrypt a Connectivity Association (CA).

A CA represents a group of nodes connected through a unidirectional secure channel. It is a security relationship, established and maintained by key agreement protocols comprised of a fully connected subset of the service access points. These access points are in stations attached to a single LAN that are supported by MACsec. Each CA has its own Security Association (SA) that uses its own SAK. More than one SA is permitted in a CA for the purpose of a key change without traffic interruption, but at least two SAs are required for rollover.

MACsec security TAG

The MACsec 802.1AE header includes a security TAG (SecTAG) field that contains the following information:

- association number within the channel
- packet number to provide a unique initialization vector for encryption and authentication algorithms, as well as protection against replay attack
- optional LAN-wide secure channel identifier

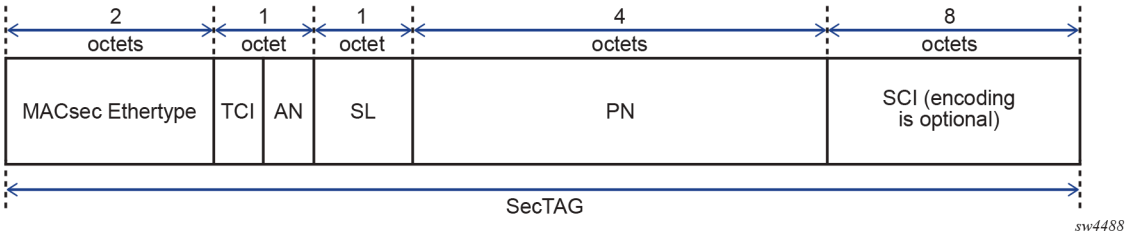
The SecTAG field, which is identified by the MACsec EtherType, conveys the following information:

- TAG Control Information (TCI)
- Association Number (AN)

- Short Length (SL)
- Packet Number (PN)
- Optionally-encoded Secure Channel Identifier (SCI)

The following figure shows the format of the SecTAG.

Figure 5: SecTAG format



17.2 MACsec and VLAN tags in clear

The main modes of encryption in MACsec are:

- VLAN in clear text (WAN mode)
- VLAN encrypted (LAN mode)

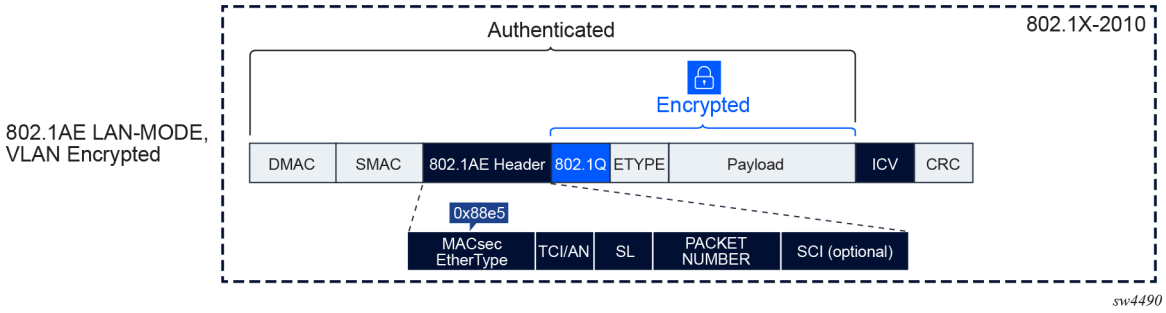
MACsec uses 802.1AE standard to encrypt the Layer 2 packet including the 802.1Q tag and the payload.

With 802.1AE encryption, the following apply to MACsec:

- frames are encrypted and protected with an integrity check value (ICV)
- MACsec EtherType is 0x88e5
- there is no impact to IP MTU or fragmentation

The following figure shows the MACsec LAN mode structure.

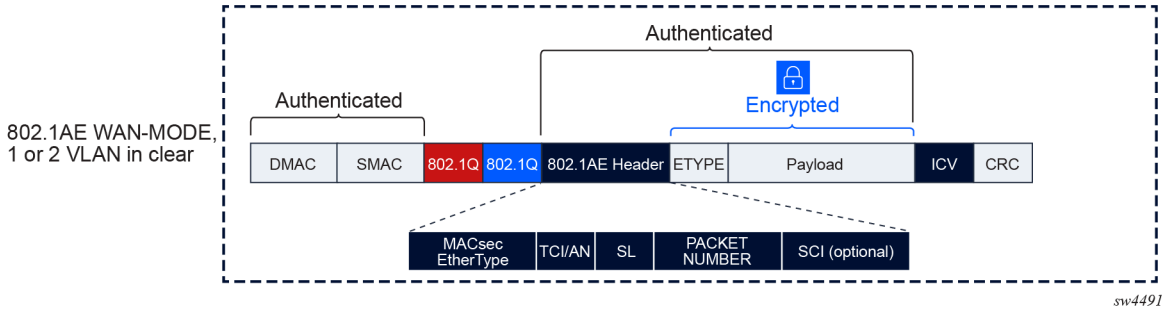
Figure 6: MACsec LAN mode structure



The original 802.1AE encrypts the 802.1Q and QinQ tags. In a WAN environment where the switching could be based on 802.1Q tag, Nokia recommends putting the 802.1Q in clear.

The following figure shows the MACsec WAN mode structure.

Figure 7: MACsec WAN mode structure



MACsec can be configured to leave one or two VLAN tags in clear. When MACsec is configured to put a single VLAN tag in clear, MACsec blindly puts 4 bytes in clear, regardless of whether there is a VLAN present. The same is true when two VLAN tags are in clear; when this occurs, MACsec blindly puts 8 bytes in clear.

Note: On SR Linux, MACsec can only be configured at the interface level with VLAN tags in clear. VLAN tags can be put in clear for hashing purposes, but SR Linux does not support MACsec at the VLAN or subinterface levels.

17.3 MACsec encryption offset

When enabling encryption, MACsec can set an offset that sends the first 30 or 50 octets in unencrypted plain text.

When the offset is set to 30, the IPv4 and TCP and UDP headers are left unencrypted while the rest of the traffic is encrypted. When the offset is set to 50, the IPv6 and TCP and UDP headers are left unencrypted while the rest of the traffic is encrypted.

The MACsec encryption offset is typically used for forward traffic when a feature requires data to be expressed in octets to perform a function like load balancing (that is, where the IP and TCP and UDP headers in the first 30 or 50 octets need to be in clear text to properly load balance traffic).

The default offset is 0, so all traffic on the link is encrypted when the encryption option is enabled and an offset is not set.

17.4 MACsec pre-shared key and keychain

MACsec uses pre-shared key (PSK) to wrap the SAK. SAKs are generated and distributed by the key server in CA to encrypt data plane PDUs. The key server on the CA is negotiated through MACsec Key Agreement (MKA) protocol based on a configurable key server value.

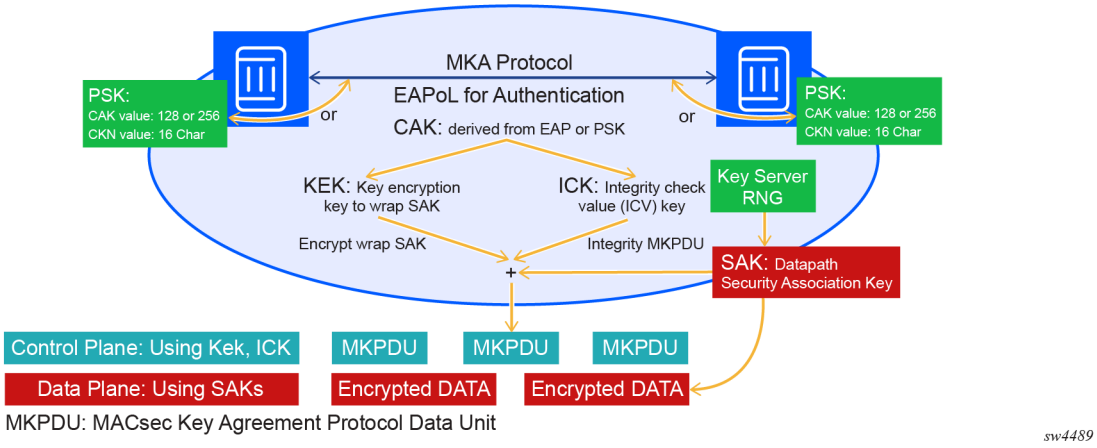
PSK is built using the following:

- Connectivity Association Name (CKN), a 64-digit hexadecimal number
- Connectivity Association Key (CAK), a 32-digit hexadecimal number

On SR Linux, the PSK is configured using keychains and can be rolled over based on the keychain lifetime. The CAK and CKN can only be changed in the key entry of a keychain when the key entry is administratively down.

The following figure shows how MACsec uses the CAK to secure the SAK to be distributed from the key server to the peer.

Figure 8: MACsec generating the CAK



After the CAK is configured in the keychain it is used to obtain the following keys:

- KEK (Key Encryption Key), which is used to wrap encrypt the SAKs
- ICK (Integrity Connection Value (ICV) Key), which is used to check the integrity of each MKPDU send between two CAs

The key server then generates a SAK that is shared with the CAs of the security domain, and that SAK is used to secure all data traffic traversing the link. The key server periodically creates and shares a randomly created SAK over the point-to-point link for as long as MACsec is enabled.

The following process occurs when enabling MACsec using a keychain:

1. You can configure multiple entries in the keychain that becomes active in certain time and date. The **send-lifetime** value of the entry is when the entry becomes active. Two entries should not have the same **send-lifetime** value.
   Each entry has a CKN and CAK that should be programmed when the entry is administratively down. The CKN and CAK must match on all CA participants at that date and time. Nokia does not recommend configuring the same CKN on multiple keychain entries.
2. The entry is activated based on its **send-lifetime** value.
3. The entry creates a new MKA session and secures the SAK with its CAK, as described in this section.
4. Based on the key server configuration of the entry, one peer in the CA becomes the key server.
5. The key server then creates a randomized SAK that is shared only with the other device over the MACsec-secured link.
6. The key server generates other MKPDUs.
7. All MKPDUs and SAKs are transmitted to other CAs using integrity check and encryption:
   - The integrity check is done through ICK

- The encryption is done through KEK
- ICK and KEK are generated from CAK using KDF (Key Derivation Function)

The KEK is used by MKA to securely transport a succession of SAKs for use by MACsec to other members of a CA. It is derived from the CAK using the transformations described in the following table.

Table 21: KEK definitions

KEK	CAK derivatives and definition
KEK	KDF (Key, Label, Key ID, KEKLength)
Key	CAK
Label	IEEE8021 KEK
Key ID	The first 16 octets of the CKN, with null octets appended to pad to 16 octets if necessary
KEKLength	Two octets representing an integer value (128 for a 128 bit KEK, 256 for a 256 bit KEK) with the most significant octet first

The ICK is used to verify the integrity of MKA PDUs and is derived from the CAK using the transformations described in the following table.

Table 22: ICK definitions

ICK	CAK derivatives and definition
ICK	KDF (Key, Label, Key ID, ICK)
LengthKey	CAK
Label	IEEE8021 ICK
Key ID	The first 16 octets of the CKN, with null octets appended to pad to 16 octets if necessary
ICKLength	Two octets representing an integer value (128 for a 128 bit ICK, 256 for a 256 bit ICK) with the most significant octet first

Note: The randomized security key enables and maintains MACsec on the point-to-point link. The key server periodically creates and shares a randomly created security key over the point-to-point link for as long as MACsec is enabled.

Configurable MKPDU destination MAC

SR Linux supports the ability to explicitly set the destination MAC address of the MKA packet to match a the unicast destination MAC address of the peer. This capability ensures that MKA packets use unicast addressing through the Layer 2 network between the two peers rather than the multicast MAC address assigned to MKPDUs.

When the unicast MAC address is configured, only the peer router identified by the destination MAC address examines the packet for MKA establishment, reducing the network noise from MKPDUs.

17.5 MACsec keychain considerations

When configuring a MACsec keychain, Nokia recommends that key entries in a keychain:

- should not have the same start time
- should have a time distance of at least one minute
- should not have the same CKN

The following shows an example of a keychain configuration.

Example: Keychain configuration

```
--{ * candidate shared default }--[ system authentication keychain AABBCCDD ]--
A:root@srl2# info with-context
    system {
        authentication {
            keychain AABBCCDD {
                admin-state enable
                type macsec
                key 1 {
                    algorithm aes-256-cmac
                    macsec {
                        cak $aes1$AWbFjByk7B2lWG8=$vzHM3n5itXoYRc2CceaTCpQv8QmKRlCiXO9Nne
ByprL/T71keYsutp9SWHwT3ZLOuBLbSzJT0jLfhK6zgq6KVg==
                        key-name
 AABBCCDDEEFF0000000000000000000000000000000000000000000000000001
                        admin-state enable
                    }
                    send-lifetime {
                        start-time 2025-02-21T21:00:00.000Z
                    }
                }
                key 2 {
                    algorithm aes-256-cmac
                    macsec {
                        cak $aes1$AWZ90wmhzvYqFm8=$nkij57SBGanMvRjrLkXy0PM9tIG5BQ
+boT0e9qGC+u3oLV5ZkeLWJHxC1Yq/gvl+4cBfwbyYru4vznc9xU2AfA==
                        key-name
 AABBCCDDEEFF0000000000000000000000000000000000000000000000000002
                        admin-state enable
                    }
                    send-lifetime {
                        start-time 2025-02-21T21:01:00.000Z
                    }
                }
            }
        }
    }
```

SR Linux uses keychain for PSK rotation. The following describes MACsec keychain considerations:

- Keychain type must be set to MACsec.
- The **receive-lifetime** command is not functional for the MACsec keychain type.
- The **send-lifetime** command may be used when the key entry becomes active.
- If there is only a single key entry, the key entry life time value is automatically set to infinity.

- To activate a key entry for MACsec, the **send-lifetime** command should be set to the current time under **system time** and the **key-entry macsec** command should be administratively enabled.
- The **key-entry macsec** command is disabled by default. Furthermore, the **key-entry macsec** command context cannot be deleted when it is administratively enabled:
  - If **key-entry macsec** is disabled but the key entry is active, the MKA goes down, causing MACsec to go down. When MACsec is operationally down, it does not send traffic in clear text, and all PDUs are blackholed.
  - The CAK and the CAK name cannot be changed or removed while **key-entry macsec** is administratively enabled.
  - When the key entry is active and the **send-lifetime** is deleted or modified, the new **send-lifetime** does not take effect in the key entry. That is, if the new **send-lifetime** of the first entry is configured beyond the **send-lifetime** of the second key entry, the new **send-lifetime** of the first key entry does not affect the second key entry activation.

17.6 MACsec protocol exclusion

MACsec can exclude the following CPU-generated protocols from encryption and decryption:

- PTP
- LACP
- LLDP

Note: Some SR Linux platforms (for example, 7250 IXR Gen 2c+ platforms) cannot exclude CDP, EFM-OAM, ETH-CFM, PTP, and uBFD from the MACsec encryption and decryption process. Furthermore, MACsec cannot exclude transiting protocols or frames generated from the datapath ASIC (mostly OAM) from encryption.

17.7 MACsec keychain fallback

MACsec keychain fallback provides a secondary or backup keychain to ensure continuous encryption and the integrity of network traffic in cases where the primary keychain fails or is enabled. This fallback keychain allows the MACsec session to remain active, preventing service disruptions when a primary keychain PSK mismatch occurs, a key expires, or in any other situation where an issue that interrupts the operation of the primary keychain arises.

When a MACsec session fails due to a PSK key (CAK) or key name (CKN) mismatch or expiry, the system can automatically switch to the fallback keychain, enabling the session to continue operation. The fallback keychain has a single entry that is always active (called infinity life). The MACsec fallback keychain is optional.

The following outlines the MACsec fallback keychain process on SR Linux when a primary keychain is configured with multiple key entries, and a fallback keychain is configured with a single key entry and an infinity life:

1. MACsec uses the primary keychain and walks through the key entries at the begin time of each entry.
2. After the begin time of a key entry is triggered, the following events prompt the system to use the fallback keychain:

- the new key entry is not capable of establishing an MKA session and the previous key entry times out after 20 seconds
- the MKA key entry becomes operationally down

3. The fallback keychain becomes the active principal MKA session, distributing a new SAK to be used until the primary keychain establishes a valid MKA session again. When the primary keychain establishes the MKA again, the fallback keychain becomes the backup MKA session, and the primary keychain MKA becomes the active principal encrypting MKA session.

Note: The fallback keychain's key entry establishes an MKA session when the key entry's start time is triggered and is always established with an infinity life, but this MKA session does not generate a SAK until it becomes the principal MKA session.

Note: Nokia recommends configuring a single key entry for the fallback keychain and ensuring it uses the same CAK and CKN on both ends. Having a single key entry ensures that the fallback keychain is always established and does not switch to a new entry with an incompatible CAK, CKN, or begin time that may cause issues with the MKA session. Nokia recommends establishing and verifying the fallback keychain prior to bringing up the primary keychain.

To configure MACsec fallback keychain on SR Linux, use the **transport-security macsec interface mka fallback-key-chain** command. The following shows an example MACsec fallback keychain configuration.

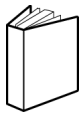
Example: MACsec fallback keychain configuration

```
--{ + candidate shared default }--[ system authentication ]--
A:root@srl2# info with-context
    keychain aabbccddeeff001 {
     admin-state enable
        type macsec
        key 1 {
            algorithm aes-128-cmac
            macsec {
                cak $aes1$AWbF/HhcXltzsm8=$J5U/4xwpX5x0y7Rb0Ux4tlbN3oIlYaMu3OfmCHKYRdM=
                key-name aabbccddeeff0011aabbccddeeff0011
                admin-state enable
            }
            send-lifetime {
                start-time 2026-01-05T13:46:42.730Z
                send-and-receive true
            }
        }
    }
    keychain kc1 {
type macsec-fallback
        key 1 {
            algorithm aes-128-cmac
            macsec {
                cak $aes1$AWbbKZtQRJCXQ28=$m7qUUZBD2y4C1Ue87W1Ux3ErR6/qJe6amhzNUHl5B8Y=
                key-name aabbccddeeff0011aabbccddeeff0013
            }
        }
    }

--{ + candidate shared default }--[ transport-security macsec interface ethernet-1/1
 mka ]--
A:root@srl2# info with-context
    transport-security {
        macsec {
```

```
            interface 1 {
                mka {
                    mka-policy 1
                    key-chain AABBCCDD
                    fallback-key-chain WWXXYYZZ
                }
            }
        }
    }
```

Customer document and product support



Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback