7705 Service Aggregation Router Gen 2

Release 26.3.R1

System Management Guide

3HE 29571 AAAA TQZZA 01
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

List of tables

List of figures

1 Getting started

1.1 About this guide

This guide describes system concepts and provides configuration explanations and examples to configure SR OS boot option file (BOF), file system and system management functions.

This guide is organized into functional chapters and provides concepts and descriptions of the implementation flow, as well as Command Line Interface (CLI) syntax and command usage.

Unless otherwise indicated, the topics and commands described in this guide apply only to the 7705 SAR Gen 2 platforms listed in Platforms and terminology.

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

Note: Unless otherwise indicated, CLI commands, contexts, and configuration examples in this guide apply for both the classic CLI and the MD-CLI.

The SR OS CLI trees and command descriptions can be found in the following guides:

- *7705 SAR Gen 2 Classic CLI Command Reference Guide*
- *7705 SAR Gen 2 Clear, Monitor, Show, Tools CLI Command Reference Guide* (for both the MD-CLI and classic CLI)
- *7705 SAR Gen 2 MD-CLI Command Reference Guide*

Note: This guide generically covers Release 26.*x*.R*x* content and may contain some content that will be released in later maintenance loads. See the *SR OS R26.x.Rx Software Release Notes*, part number 3HE 29176 000*x* TQZZA, for information about features supported in each load of the Release 26.*x*.R*x* software. For a list of features and CLI commands that are present in SR OS but not supported on the 7705 SAR Gen 2 platforms, see "SR OS Features not Supported on SAR Gen 2" in the *SR OS R26.x.Rx Software Release Notes*.

1.2 Platforms and terminology

Note:
Unless explicitly noted otherwise, this guide uses the terminology defined in the following table to collectively designate the specified platforms.

Table 1: Platforms and terminology

Platform	Collective platform designation
7705 SAR-Hx	7705 SAR Gen 2
7705 SAR-Mx	

Platform	Collective platform designation
7705 SAR-1	

1.3 Node management using VPRN

While customarily node management is operated either via the out-of-band interface or in-band via the Base routing instance, it is also possible to manage the node using a VPRN. Both IPv4 and IPv6 are supported.

The following management plane clients are supported using VPRN:

- DNS
- gRPC (dial-out telemetry)
- RADIUS
- SNMP (traps)
- SSH
- Syslog
- TACACS+
- Telnet

The following servers are supported using VPRN:

- FTP
- gRPC
- NETCONF (including notifications)
- SNMP
- SSH
- Telnet

For more details, see the *7705 SAR Gen 2 Layer 3 Services Guide: IES and VPRN*, "*Node management using VPRN*".

1.4 Conventions

This section describes the general conventions used in this guide.

1.4.1 Precautionary and information messages

The following information symbols are 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.

1.4.2 Options or substeps in procedures and sequential workflows

Options in a procedure or a sequential workflow are indicated by a bulleted list. In the following example, at step 1, the user must perform the described action. At step 2, the user must perform one of the listed options to complete the step.

Example: Options in a procedure

1. User must perform this step.
2. This step offers three options. User must perform one option to complete this step.
   - This is one option.
   - This is another option.
   - This is yet another option.

Substeps in a procedure or a sequential workflow are indicated by letters. In the following example, at step 1, the user must perform the described action. At step 2, the user must perform two substeps (a. and b.) to complete the step.

Example: Substeps in a procedure

1. User must perform this step.
2. User must perform all substeps to complete this action.
   a. This is one substep.
   b. This is another substep.

Nested substeps within a procedure or a sequential workflow are indicated by roman numerals. In the following example, at step 1, the user must perform the described action. At step 2, the user must perform two substeps (a. and b.) to complete the step. At substep b, the user must perform two additional substeps (i. and ii.) to complete the step.

Example: Nested substeps in a procedure

1. User must perform this step.
2. User must perform all substeps to complete this action.
   a. This is one substep.
   b. User must perform all nested substeps to complete this action.

i. This is a nested substep.
ii. This is another nested substep.

2 Security

This chapter provides information to configure security features.

2.1 Authentication, authorization, and accounting

This chapter describes authentication, authorization, and accounting (AAA) used to monitor and control network access on routers. Network security is based on a multistep process. The first step, authentication, validates a user's credentials. The second step, authorization, allows the user to access and execute commands at various command levels based on profiles assigned to the user.

The third step, accounting, keeps track of the activity of users who have accessed the network. The type of accounting information recorded can include a history of the commands executed, the amount of time spent in the session, the services accessed, and the data transfer size during the session. The accounting data can be used for trend analysis, billing, and auditing purposes.

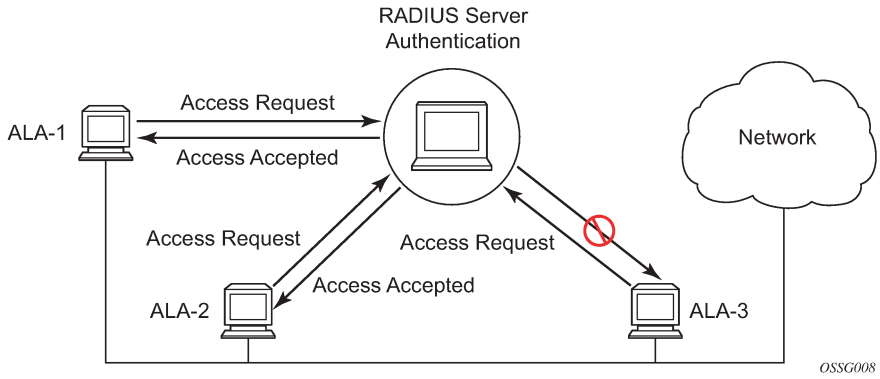
The router can be configured to use local, Remote Authentication Dial-In User Service (RADIUS), Terminal Access Controller Access Control System Plus (TACACS+), or Lightweight Directory Access Protocol (LDAP) security to validate users who attempt to access the router by console, Telnet, File Transfer Protocol (FTP), or other supported interfaces. Users can also select the order in which the authentication methods are attempted.

The router supports the following security features:

- Local security can be implemented for authentication and authorization.
- RADIUS can be used for authentication, authorization, and accounting in the Base routing instance or a VPRN.
- TACACS+ can be used for authentication, authorization, and accounting in the Base routing instance or a VPRN.
- LDAP can be used for authentication in the Base routing instance.

The following figure shows end user access-requests sent to a RADIUS server. After validating the usernames and passwords, the RADIUS server returns an access-accept message to the users on ALA-1 and ALA-2. The username and password from ALA-3 could not be authenticated; therefore, access was denied.

Figure 1: RADIUS requests and responses



2.1.1 Authentication

Authentication validates a user's credentials when a user attempts to log in.

When a user attempts to log in through the console, FTP, or other methods, the client sends credentials to the router. Based on the received credentials, the router creates and sends an authentication request to a RADIUS, TACACS+, LDAP, or local database. The order in which the router tries different types of AAA servers and local databases is defined by the configured authentication order.

Transactions between the router and a RADIUS or TACACS+ server are authenticated through the use of a shared secret. The secret is never transmitted over the network. TLS can be used for the connection between the router and the LDAP or RADIUS server. User passwords are sent encrypted between the client and the AAA (RADIUS, TACACS+, or LDAP) server which prevents someone snooping on an insecure network to learn password information.

If the AAA server (of the chosen authentication method) does not respond within a specified time, the router issues the access request to the next configured servers of the same authentication method. Each AAA server must be configured identically to guarantee consistent results.

If any AAA server rejects the authentication request, it sends an access reject message to the router. In this case, no access request is issued to any other AAA servers of the chosen authentication method. However, if other authentication methods, such as TACACS+ or local, are configured and the **exit-on-reject** option is not set, then these methods are attempted. If no other authentication methods are configured, or all methods reject the authentication request, then access is denied.

For the AAA server selection, round-robin is used if multiple AAA servers for one particular authentication method are configured. Although, if the first alive server in the list cannot find a username, the router does not re-query the next server in the AAA server list for that authentication method and denies the access request. It may get authenticated on the next login attempt if the next selected AAA server has the appropriate username. It is recommended that the same user databases are maintained for AAA servers to avoid inconsistent behavior.

The user login is successful when the AAA server accepts the authentication request and responds to the router with an access accept message.

Implementing authentication without authorization for the routers does not require the configuration of VSAs (Vendor Specific Attributes) on the RADIUS server. However, users, user access permissions, and command authorization profiles must be configured on each router.

Any combination of these authentication methods can be configured to control network access from a router:

- Local authentication
- RADIUS authentication
- TACACS+ authentication
- LDAP authentication

Note:
Multi-factor authentication (MFA) is not supported for local users, but is supported with RADIUS and TACACS+ servers that provide MFA functionality for remote users.

2.1.1.1 Local authentication

Local authentication uses PKI or usernames and passwords as authentication credentials to authenticate login attempts. The authentication credentials are local to each router, not to user profiles.

By default, local authentication is enabled. When one or more of the other security methods are enabled, local authentication is used in case it is configured as first method in the authentication order, or if other authentication methods are configured before local in the authentication order and fail.

Locally, usernames, public keys, and password management information can be configured. This is referred to as local authentication.

2.1.1.2 RADIUS authentication

RADIUS is a client/server security protocol and software that enables remote access servers to communicate with a central server to authenticate users and authorize access to the requested system or service.

RADIUS allows administrators to maintain user profiles in a shared central database and provides better security, allowing a company to set up a policy that can be applied at a single administered network point.

2.1.1.2.1 RADIUS server selection

The RADIUS server selection algorithm is used by different applications:

- RADIUS for user management
- RADIUS authentication for Enhanced Subscriber Management
- RADIUS accounting for Enhanced Subscriber Management

Up to five RADIUS servers can be configured for each application.

The RADIUS server selection algorithm uses two modes, direct mode or round-robin mode.

- **direct mode**
  The first server is used as the primary server. If this server is unreachable, the next server based on the server index is used. This continues until either all servers have been tried or an answer is received.
- **round-robin mode**
  The RADIUS application sends the next RADIUS packet to the next server.

A RADIUS server is considered available initially and marked as unavailable if no response packets are received in a period equal to the configured packet timeout multiplied by the retry count after sending a request.

A server that is down can only be tried again after 30 seconds have elapsed. If, within the 30 seconds, a valid RADIUS reply is received for that server, the server immediately becomes available again.

A server is always marked as available when any valid RADIUS packet is received from that server.

Use the following commands to configure the RADIUS server selection algorithm:

- **MD-CLI**

```
configure aaa radius server-policy servers access-algorithm
configure service vprn aaa remote-servers radius access-algorithm
configure system security aaa remote-servers radius access-algorithm
```

- **classic CLI**

```
configure aaa radius-server-policy servers access-algorithm
configure service vprn aaa remote-servers radius access-algorithm
configure system security radius access-algorithm
```

2.1.1.2.1.1 RADIUS challenge/response interactive authentication

Challenge/response interactive authentication is used for multi-factor authentication (MFA) where the RADIUS server asks for response to a displayed challenge. The challenge packet includes a challenge to be displayed to the user that is sent by the RADIUS server, such as a unique generated numeric value unlikely ever to be repeated.

The user then enters the challenge into his device (or software) and it calculates a response, which the user enters into the client which forwards it to the RADIUS server within an access request. If the response matches the expected response, the RADIUS server allows the user access, otherwise it rejects the response.

Use the following command to enable RADIUS challenge/response mode:

- **MD-CLI**

```
configure system security aaa remote-servers radius interactive-authentication
```

- **classic CLI**

```
configure system security radius interactive-authentication
```

RADIUS interactive authentication is disabled by default. Enabling interactive authentication does not mean that the system uses RADIUS challenge/response mode by default. The configured authentication order is used. If the authentication order is local and then RADIUS, the system will first attempt to login the user using local authentication. If this fails, the system will revert to RADIUS and challenge/response mode.

If the authentication order is RADIUS and then local, the standard password prompt is always displayed. The user enters a username and password at this prompt. If RADIUS interactive authentication is enabled the password does not have to be the correct password because authentication is accomplished using the RADIUS challenge/response method. The user can enter any password. The username and password are sent to the RADIUS server, which responds with a challenge request that is transmitted back to the node

by the RADIUS server. When the user enters the challenge response, the response is authenticated by the RADIUS server to allow node access to the user.

For example, if the system is configured with system security authentication-order set to local and then RADIUS, at the login prompt the user can enter the username "admin" and the corresponding password. If the password for local authentication does not match, the system falls into RADIUS authentication mode. The system checks the interactive-authentication configuration and if it is enabled it enters into challenge/response mode. It sends the username and password to the RADIUS server, and the server sends the challenge request back to the node and to the user where it appears as a challenge prompt on screen. A challenge received from the RADIUS server typically contains a string and a hardware token that can be used to generate a password on the users' local personal token generator. For example, the RADIUS server may send the challenge prompt "Enter response for challenge 12345:" to the SR OS. The string "12345" can be entered in the local token generator which generates the appropriate challenge response for the entered string. This challenge response can then be entered on the SR OS prompt for authorization.

When the user enters the correct challenge response they are authenticated using the RADIUS server.

If session timeout and idle timeout values are configured on the RADIUS server, these are used to determine the length of time before the SR OS cancels the challenge prompt. If the user is idle longer than the received idle timeout (seconds) from the RADIUS server, and/or if the user does not press **Enter** before the received session timeout (seconds).

Note: For SSH only the session timeout value is used. The SSH application cannot track character input into the login prompt until **Enter** is pressed.
If the idle or session timeout is not available or if the value is set to a very large number, the SR OS uses the smallest value set with the **configure system login-control idle-timeout** command and the idle/session timeout value to terminate the prompt. If the idle timeout is disabled, the maximum idle timeout (24 hours) is used for the calculation.

The SR OS displays the login attempts/failure per user in the "show system security user username" screen. If the RADIUS rejects a challenge response, it counts as a failed login attempt and a new prompt is displayed. The number of failed attempts is limited by the value set by the following commands:

- **MD-CLI**

```
configure system security user-params attempts
```

- **classic CLI**

```
configure system security password attempts
```

An incorrect challenge/response results in a failure count against the password attempts.

2.1.1.2.1.2 Server reachability detection

A server is reachable when the server is operationally up and a valid response is received within a timeout period that is configured using the retry command option at the RADIUS policy level.

A server is considered unreachable when the server is operationally down, or when one of the following situations occurs:

- **timeout**

  A number of consecutive timeouts are encountered for a specific server. Use the following commands to configure the timeout:

– **MD-CLI**

```
configure aaa radius server-policy servers timeout
configure service vprn aaa remote-servers radius server-timeout
configure system security aaa remote-servers radius server-timeout
```

– **classic CLI**

```
configure aaa radius-server-policy servers timeout
configure service vprn aaa remote-servers radius timeout
configure system security radius timeout
```

- **send failure**

  A packet cannot be sent to the RADIUS server because the forwarding path toward the RADIUS server has failed; for example, the route is not available or the interface is shut down.

  If a send failure occurs, no retry mechanism is invoked and the next server in line is immediately used.

A server that is down can only be used again after 30 seconds have elapsed. If, within the 30 seconds, a valid RADIUS reply is received for that server, the server immediately becomes available again.

The operational state of a server can also be "unknown" if the RADIUS application is not aware of the state of the RADIUS server. For example, if the server was previously down but no requests had been sent to the server, it would not be certain yet whether the server is reachable.

2.1.1.3 TACACS+ authentication

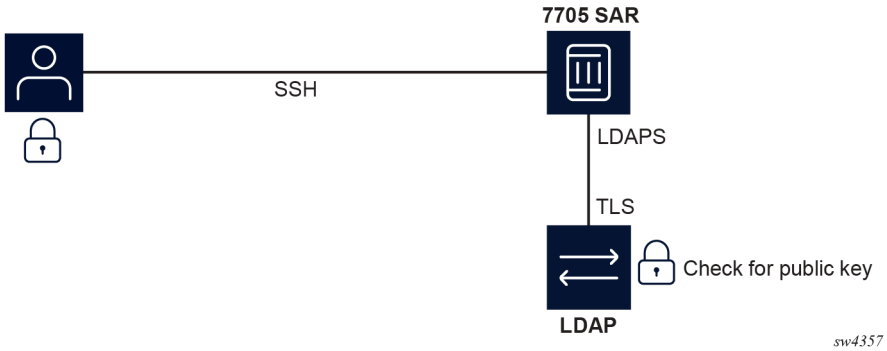
Terminal Access Controller Access-Control System Plus (TACACS+) is an authentication protocol that allows a remote access server to forward a user login password to an authentication server, to determine whether access is allowed to a system. In contrast to RADIUS, which combines authentication and authorization, TACACS+ separates these operations.

2.1.1.4 LDAP authentication

Lightweight Directory Access Protocol (LDAP) can provide authentication, authorization, accounting (AAA) functionality, and can allow users to access the full virtualized data center and networking devices. SR OS currently supports LDAP provision of a centralized authentication method with public key management. The authentication method is based on SSH public keys or keyboard authentication (username, password).

Administrators can access networking devices with one private key; public keys are usually saved locally on the SSH server. Proper key management is not feasible with locally-saved public keys on network devices or on virtual machines, as this would result in hundreds of public keys distributed on all devices. LDAPv3 provides a centralized key management system that allows for secure creation and distribution of public keys in the network. Public keys can be remotely saved on the LDAP server, which makes key management much easier, as shown in Figure 2: Key management.

Figure 2: Key management



The administrator starts an SSH session through an SSH client using their private key. The SSH client for the authentication method sends a signature created with the user's private key to the router. The router authenticates the signature using the user's public key and gives access to the user. To access the public key, the router looks up the public key stored on the LDAP server and the public key stored locally. The order in which the public keys are looked up is defined by the authentication order. Communication between the router and the LDAP server should be secured with LDAP over SSL/TLS (LDAPS). After successfully opening a secured connection, LDAP returns a set of public keys that can be used by the router to verify the signature.

LDAP is integrated into the SR OS as an AAA protocol alongside existing AAA protocols, such as RADIUS and TACACS+. The AAA framework provides tools and mechanisms (such as method lists, server groups, and generic attribute lists) that enable an abstract and uniform interface to AAA clients, irrespective of the actual protocol used for communication with the AAA server.

The authentication functions are:

- **public key authentication**

  The client tries to SSH to the SR OS using public keys.

  Public keys can be stored locally or on the LDAP server and retrieved as needed to authenticate the user.

- **password authentication (keyboard interactive)**

  The LDAP server can be used for user authentication using keyboard interactive, as with simple username and password authentication.

2.1.1.4.1 LDAP authentication process

A client starts an LDAP session by connecting to an LDAP server, called a Directory System Agent (DSA), which–by default–are on TCP port 389 and UDP port 636 for LDAP. The SR OS then sends an operation request to the server, and the server sends responses in return, as shown in Figure 3: LDAP server and SR OS interaction for retrieving the public key. With some exceptions, the client does not need to wait for a response before sending the next request, and the server may send the responses in any order. All information is transmitted using Basic Encoding Rules (BER).

In the SR OS, the client can request the following operations:

- **StartTLS**

  Uses the LDAPv3 Transport Layer Security (TLS) extension for a secure connection.

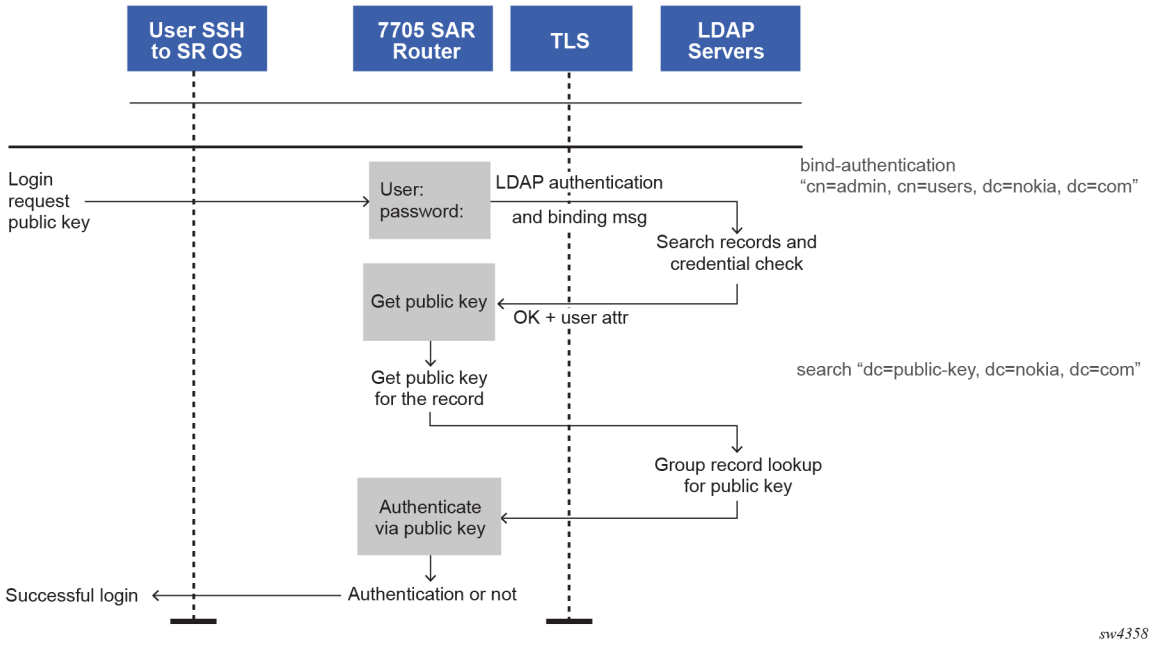
- **Bind**

  Authenticates and specifies the LDAP protocol version.
- **Search**

  Searches for and retrieves directory entries.
- **Unbind**

  Closes the connection (not the inverse of Bind).

Figure 3: LDAP server and SR OS interaction for retrieving the public key



The connection between the router as the LDAP client and the LDAP server should be encrypted using TLS, as all credentials between the router and LDAP are transmitted in clear text.

2.1.1.4.2 Authentication order

SR OS supports local and LDAP public key storage, the order of which is configurable. Use the following command to configure authentication order:

- **MD-CLI**

```
configure system security user-params authentication-order
```

- **classic CLI**

```
configure system security password authentication-order
```

The SR OS sends available authentication methods to the client and supports public key and password authentication. Use the following command to configure the client to use the public key authentication method:

- **MD-CLI**

```
configure system security aaa remote-servers ldap public-key-authentication
```

- **classic CLI**

```
configure system security ldap public-key-authentication
```

If the client chooses the public key and LDAP is first in authentication order, the SR OS tries to authenticate using public key retrieval from the LDAP server. If the public key retrieval from LDAP server fails and **exit-on-reject** is not configured, the SR OS tries the next method (**local**) in the authentication order for the public key. If the next method also fails, a user authentication fail message is sent to the client.

If the public key retrieval from the LDAP server fails and **exit-on-reject** is configured, the SR OS does not try the next method in the authentication order. A user-authentication fail message is sent to the client. At this point, the client can be configured to only use public key authentication or use both public key authentication followed by password authentication. If the client is configured to use password authentication, it goes through the authentication order again (for example, it tries all the configured methods in the configured authentication order) as long as **exit-on-reject** is not configured.

2.1.1.4.2.1 Authentication order public key detail

There are two keys for public key authentication: a private key stored on the client and a public key stored on the server (local) or AAA server (LDAP). The client uses the private key to create a signature, which only the public key can authenticate. If the signature is authenticated using the public key, then the user is also authenticated and is granted access. SR OS can locally store, using CLI, as many as 32 RSA keys and 32 ECDHA keys for a single user. In total, the SR OS can load a maximum of 128 public keys in a single authentication attempt.

Note: The client creates a signature using a single private key, but this signature can be authenticated on the SR OS with maximum of 128 public keys in a single try. If all these public keys fail to authenticate, then a failure message is sent to the client and the number of failed attempts is incremented.

If the client has another private key, it can create a new signature with this new private key and attempt the authentication one more time, or switch to password authentication.

The following steps describe the procedure where the client attempts to authenticate using a public key and the authentication order is configured as **ldap**, then **local**.

Note: With each increment of failed attempts, the SR OS also checks the limit for lock-out. If the limit is reached, the user is locked out.

1. The SSH client opens a session and tries to authenticate the user with private-key-1 (creating signature-1 from private-key-1).
2. The SR OS checks the authentication order.
3. The SR OS loads public keys for the user, as follows.
   a. If **exit-on-reject** is not configured, the SR OS loads all public keys from the LDAP server and all public keys from the locally-saved location.
   b. If **exit-on-reject** is configured, the SR OS only loads all public keys from the LDAP server and not from the locally-saved location.

4. The SR OS compares received client signature-1 with signature calculated from loaded public keys and attempts to find a match.
   a. If a match is found, the user is authenticated. The procedure ends.
   b. If no match is found, authentication fails and the SSH client is informed. The LDAP server waits for the SSH client's reaction.
5. The SSH client reacts in one of several ways.
   a. The connection is closed.
   b. The password authentication method is continued. In this case, on the SR OS, the number of failed authentication attempts is not incremented.
   c. The next public key is continued, as follows.
      i. If it is not 21st received public key, return to step 3.
      ii. If it is the 21st received public key, the number of failed authentication attempts is incremented and the connection is closed.

2.1.1.4.3 LDAP authentication using a password

In addition to public key authentication, the SR OS supports password (keyboard) authentication using the LDAP server.

Note: TLS provides the encryption for password authentication.

In the following example, the client attempts to authenticate using a password and only LDAP is configured in the authentication order.

1. The client uses Telnet or SSH to reach the SR OS.
2. The SR OS retrieves the username and password (in plain text).
3. The SR OS performs a bind operation to the LDAP server. Use the following command to set the root DN and password:
   - **MD-CLI**

     ```
     configure system security aaa remote-servers ldap server bind-authentication password
     configure system security aaa remote-servers ldap server bind-authentication root-dn
     ```

   - **classic CLI**

     ```
     configure system security ldap server bind-authentication password password root-dn
     ```

4. The SR OS performs a search operation for the username on LDAP server.
   a. If the username is found, LDAP sends user_distinguished_name to the router.
   b. If the username is not found, the authentication fails. The attempt and failed attempt counters are incremented.
5. The SR OS performs a bind operation to LDAP with user_distinguished_name and the password from step 2.
6. The LDAP server checks the password.

a. If the password is correct, the bind operation succeeds. The failed attempt and successful attempt counters are incremented.

b. If the password is incorrect, bind is unsuccessful and authentication fails. The attempt and failed attempt counters are incremented.

7. The SR OS sends a message to unbind from the LDAP server.

2.1.1.4.4 Timeout and retry configuration for the LDAP server

Use the following commands to configure the number of retry attempts and the response timeout for the LDAP server:

- **MD-CLI**

```
configure system security aaa remote-servers ldap server-retry
configure system security aaa remote-servers ldap server-timeout
```

- **classic CLI**

```
configure system security ldap retry
configure system security ldap timeout
```

The server retry value is the maximum number of connection attempts that the SR OS can make to reach the current LDAP server before attempting the next server. For example, if the value is set to the default of 3, the SR OS tries to establish the connection to current server three times before attempting to establish a connection to the next server.

The server timeout value is the number of seconds that the SR OS waits for a response from the server with which it is attempting to establish a connection. If the server does not reply within the specified timeout value, the SR OS increments the retry counter by one. The SR OS attempts to establish the connection to the current server up to the configured retry value before moving to the next configured server.

2.1.1.4.5 TLS behavior and LDAP

RFC 4511 section 4.14.1 states, "A client requests TLS establishment by transmitting a StartTLS request message to the server" and "The client MUST NOT send any LDAP PDUs at this LDAP message layer following this request until it receives a StartTLS Extended response". As such, if an LDAP has a TLS profile configured and the TLS is in an operationally down state, no LDAP packets are transmitted if TLS negotiation has not been completed, including when the TLS profile is shut down.

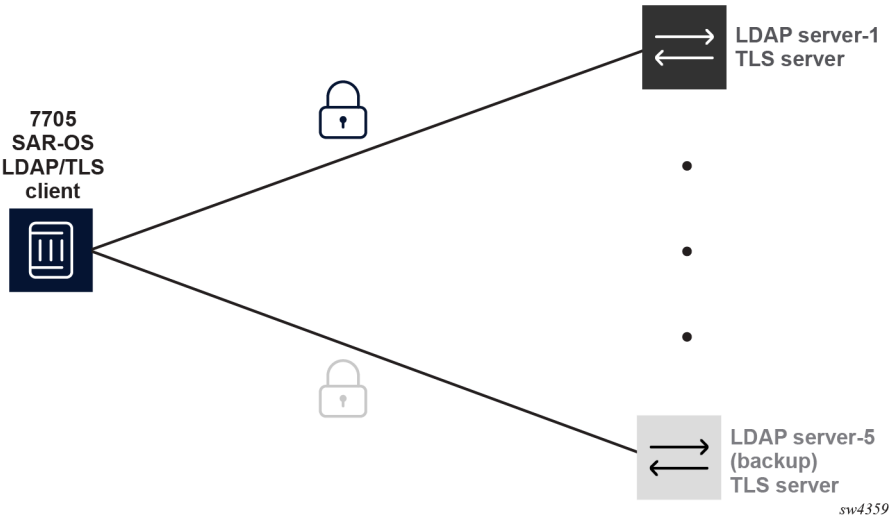
2.1.1.4.6 LDAP redundancy and TLS

LDAP supports up to five redundant (backup) servers. Depending on the configuration of **timeout** and **retry** values, if an LDAP server is found to be out of service or operationally down, the SR OS will switch to the redundant servers. The SR OS will try the next LDAP server in the serred server index.

LDAP servers can use the same TLS profile or can have their own TLS profile. Each TLS profile can have a different configuration of **trust-anchor** and **cert-profile**. For security reasons, the LDAP server could be in different geographical areas and, therefore, each will be assigned its own server certificate and trust anchor. The TLS profile design allows users to mix and match all components.

Redundant LDAP servers are shown in Figure 4: LDAP and TLS redundancy.

Figure 4: LDAP and TLS redundancy



2.1.1.5 Password hashing

SR OS supports two algorithms for user password hashing: bcrypt, which is the default algorithm, and PBKDF2. The PBKDF2 algorithm can use SHA2 (SHA-256) or SHA3 (SHA-512) for hashing.

Use the following command to configure the algorithm to hash all user passwords:

- **MD-CLI**

```
configure system security user-params local-user password hashing
```

- **classic CLI**

```
configure system security password hashing
```

When password hashing is configured, the following sequence of steps occurs at login:

1. The node checks the stored password and notes its hash algorithm.
2. The password entered by the user is hashed with the noted algorithm, and the node compares the hash with the stored user password hash.
3. If the entered and stored passwords are the same, and if the hash algorithm of the stored user password is different than the hash algorithm of the system password, the user is prompted to enter a new password two times to ensure the passwords match. The node stores this new password in the RAM, not in the system configuration file.

   To store the new password in the configuration file, an admin user must perform an **admin save** command. If the **admin save** command is not executed, on the next reboot the hash algorithm of the stored user password may be different than the system hash and the user must go through this process again from step 2.

After an upgrade to a software load that supports PBKDF2, the default password continues to be stored using the bcrypt algorithm. The following example describes the procedure to change the algorithm. In the example, the algorithm is changed to PBKDF2 and "User_name" can be any user.

1. User_name logs in and runs the **hashing** command to change the algorithm.
2. To save the algorithm change, an admin user performs an **admin save** command.
3. To store User_name's password using PBKDF2, the admin user changes User_name's password.

   From this point onward, any new user passwords or changes to existing user passwords are stored using PBKDF2.

2.1.2 Authorization

The SR OS supports local, RADIUS, and TACACS+ authorization to control the commands a user can execute . The following authorization methods can be configured for each user:

- Local authorization – uses one or more local profiles that are applied to each user.
- RADIUS authorization – uses local profiles or profiles that are constructed using VSAs from a RADIUS server. See RADIUS VSAs for more information.
- TACACS+ authorization – TACACS+ authorization uses local profiles, profiles that are constructed using VSAs from a TACACS+ server or interactive per-command authorization. See TACACS+ services and VSAs for more information.

The profiles are applied when the user logs in. If changes are made to the profiles, the user must log out and back in again for the changes to take effect.

The following table lists authorization support.

Table 2: Authorization support

Server or profile	Classic CLI	MD-CLI	NETCONF	gRPC/gNMI
LDAP	—	—	—	—
TACACS+	Yes	Yes	Yes	Yes
RADIUS	Yes	Yes	Yes	Yes
Local	Yes	Yes	Yes	Yes

Authorization profiles apply to the classic CLI, the MD-CLI, NETCONF, and gRPC/gNMI. See Authorization profiles for classic and model-driven management interfaces for more details.

2.1.2.1 Local authorization

Local authorization uses user profiles and user access information after a user is authenticated. The profiles and user access information specifies the actions the user is allowed to perform.

For more information, see Configuring local command authorization profiles.

2.1.2.2 RADIUS authorization

RADIUS authorization grants or denies access permissions for a user. Permissions include the use of FTP, Telnet, SSH (SCP and SFTP), and console access. When granting Telnet, SSH (SCP and SFTP), and console access to the router, authorization can be used to limit user access to the CLI commands and files.

When a user has been authenticated using RADIUS (or another method), the router can be configured to perform authorization. The RADIUS server can be used to:

- download the user command authorization profile to the router
- send the profile name that the router should apply to the user (these profiles must be created on each router)

If RADIUS authentication is successful and no authorization is configured for the user on the RADIUS server, local (router) authorization is attempted, if configured using the **authentication-order** command.

When authorization is configured and profiles are downloaded to the router from the RADIUS server, the profiles are considered temporary configurations and are not saved when the user session terminates.

The temporary profiles are only downloaded if the user authenticates via RADIUS. RADIUS-based authorization is not supported for users who authenticate locally or via TACACS+.

The following table lists the supported authorization configurations.

Table 3: Supported authorization configurations

	Local	RADIUS supplied profile
Locally configured user	✓	
RADIUS server configured user	✓	✓
TACACS+ server configured user	✓	

When using authorization, maintaining a user database on the router is not required. Usernames can be configured on the RADIUS server. Usernames are temporary and are not saved in the configuration when the user session terminates. Temporary user login names and their associated passwords are not saved as part of the configuration.

2.1.2.3 TACACS+ authorization

TACACS+ command authorization operates in different modes:

- Single common access controls with a local command authorization profile
- Common access controls with per-command authorization with TACACS+
- Common access controls with privilege level mapping to local profiles
- Common access controls with TACACS+ services and VSAs

2.1.2.3.1 Single common access controls with a local command authorization profile

All users who authenticate with TACACS+ use a single common TACACS+ default template with a local authorization profile that controls:

- access to the console port CLI, FTP, gRPC, LI commands, NETCONF, SCP/SFTP, SNMP, SSH CLI, and Telnet CLI
- file access
- authorization for CLI commands, NETCONF operations, and gRPC RPCs

The **use-default-template** command must be enabled. Optionally, Vendor-Specific Attributes (VSAs) defined on the TACACS+ server can control per-user access.

To use a single, common command authorization profile for TACACS+ users, the TACACS+ default template must be enabled and configured with a valid local profile. The local profile is then used for command authorization. TACACS+ **authorization** must be disabled.

Example

Use the following commands to configure a single common local command profile called "TACPLUS_USERS" for command authorization for all TACACS+ users:

- **MD-CLI**

```
configure system security aaa remote-servers tacplus use-default-template
configure system security aaa user-template user-template-name tacplus-default profile
 "TACPLUS_USERS"
delete configure system security aaa remote-servers tacplus authorization
```

- **classic CLI**

```
configure system security tacplus use-default-template
configure system security user-template tacplus_default profile "TACPLUS_USERS"
configure system security tacplus no authorization
```

See TACACS+ configuration for file access control using VSAs for examples on optionally using VSAs for file access control.

2.1.2.3.2 Common access controls with per-command authorization with TACACS+

Per-command authorization with the TACACS+ server can be used instead of using a local CLI command authorization profile.

When the **authorization** command is enabled without the **use-priv-lvl** command, each CLI command the user issues is sent to the TACACS+ server for authorization. The SR OS authorization request contains the first word of the CLI command as the value for the TACACS+ **cmd** and all the following words as a **cmd-arg**. Quotation marks are removed from quoted values and the values are seen as one **cmd** or **cmd-arg**.

Example

Use the following commands to configure common access controls with per-command authorization with TACACS+:

- **MD-CLI**

```
configure system security aaa remote-servers tacplus authorization
configure system security aaa remote-servers tacplus use-default-template
```

- **classic CLI**

```
configure system security tacplus authorization
```

```
configure system security tacplus use-default-template
```

See TACACS+ attribute value authorization example for an example of the **cmd** and **cmd-arg** attribute value (AV) pairs sent to the TACACS+ server. SR OS sends the entire CLI context in the **cmd** and **cmd-arg** attribute value (AV) pairs. See TACACS+ configuration for file access control using VSAs for examples of optionally using VSAs for file access control.

2.1.2.3.3 Common access controls with privilege level mapping to local profiles

With local **priv-lvl-map** configuration maps the user privilege level provided by the TACACS+ server to locally-configured CLI command authorization profiles when the **use-priv-lvl** command is enabled.

When the **use-priv-lvl** command is configured, the TACACS+ server returns the privilege level for each user, which SR OS maps to a local profile configured with the **priv-lvl-map** command. The local profile is then used for command authorization. If the TACACS+ server does not return a privilege level, SR OS uses the local profile configured in the TACACS+ default template for command authorization.

Example

Use the following commands to configure common access controls with privilege level mapping to local profiles:

- **MD-CLI**

```
configure system security aaa remote-servers tacplus authorization use-priv-lvl true
configure system security aaa remote-servers tacplus priv-lvl-map priv-lvl 1 user-
profile-name "LOW"
configure system security aaa remote-servers tacplus priv-lvl-map priv-lvl 7 user-
profile-name "MEDIUM"
configure system security aaa remote-servers tacplus priv-lvl-map priv-lvl 15 user-
profile-name "HIGH"
configure system security aaa remote-servers tacplus use-default-template
configure system security aaa user-template user-template-name tacplus-default profile
 "TACPLUS_USERS"
```

- **classic CLI**

```
configure system security tacplus authorization use-priv-lvl
configure system security tacplus priv-lvl-map priv-lvl 1 "LOW"
configure system security tacplus priv-lvl-map priv-lvl 7 "MEDIUM"
configure system security tacplus priv-lvl-map priv-lvl 15 "HIGH"
configure system security tacplus use-default-template
configure system security user-template tacplus_default profile "TACPLUS_USERS"
```

2.1.2.3.4 TACACS+ attribute value authorization example

For TACACS+ authorization, SR OS the CLI command in the **cmd** and **cmd-arg** attribute value (AV) pairs in the **shell** service (not shown in the output below).

Note: Command accounting logs the command as it was entered.

Example: AV pairs resulting from commands entered in the MD-CLI

```
/configure
```

```
cmd=configure

/configure system
cmd=configure
cmd-arg=system

/configure system name node-2
cmd=configure
cmd-arg=system
cmd-arg=name
```

Example: AV pairs resulting from commands entered in the classic CLI

```
/configure
cmd=configure

/configure system
cmd=configure
cmd-arg=system

/configure system name node-2
cmd=configure
cmd-arg=system
cmd-arg=name

cmd-arg=node-2
```

2.1.2.3.5 TACACS+ configuration command authorization in model-driven interfaces

Configuration command authorization using TACACS+ sends multiple requests for the same command that may be the same depending on the configuration changes. In model-driven interfaces, command authorization is required for the following changes to the candidate configuration:

- the command that was entered; for example, **system name node-2**
- the resulting configuration changes, because other elements may be modified or deleted; for example, **delete router "Base"** deletes the entire Base router configuration, and all of the deletions must be authorized

Note: If the command authorization fails, the resulting configuration changes are not authorized.

Multiple authorization requests are also sent in the following cases:

- for MD-CLI compound commands where multiple elements are changed in a single command

```
system name node-2 location Sunnyvale
system name node-2 } router router-id 10.1.1.1
```

- configuration changed by an element's YANG modeling constraints, such as "choice" or "when" statements

Example

The following example shows how setting the system name is an operation that changes one configuration element.

```
[ex:/configure]
```

```
A:admin@node-2# system name newname

# Command authorization
cmd=configure
cmd-arg=system
cmd-arg=name
# Resulting change authorization
cmd=configure
cmd-arg=system
cmd-arg=name
```

The following log examples show that the **memory** context and the **console** command are mutually exclusive, and configuring a new value deletes the existing value. The system must also implicitly authorize the deletion.

Note: Command accounting only logs the command that is entered.

Example: Existing configuration

```
[ex:/configure log log-id "42" destination]
A:admin@node-2# info
    memory {
    }
```

Example: Configuration commands

```
[ex:/configure log log-id "42" destination]
A:admin@node-2# console
```

Example: Resulting configuration

```
[ex:/configure log log-id "42" destination]
A:admin@node-2# info
    console
```

Example: Command authorization requests

```
# Command authorization for console
cmd=configure
cmd-arg=log
cmd-arg=log-id
cmd-arg=42
cmd-arg=destination
cmd-arg=console
# Resulting change authorization for console
cmd=configure
cmd-arg=log
cmd-arg=log-id
cmd-arg=42
cmd-arg=destination
cmd-arg=console
# Resulting change delete authorization for memory
cmd=configure
cmd-arg=log
cmd-arg=log-id
cmd-arg=42
cmd-arg=destination
cmd-arg=memory
```

2.1.2.4 Authorization profiles for classic and model-driven management interfaces

Authorization profiles can be configured to match commands in either the classic CLI or MD-CLI syntax. If the command syntax is the same in both CLI engineers, a match succeeds in both CLI engines. If the command syntax is different, a match only succeeds in the CLI engine where the command syntax matches.

The following table shows authorization and matches based on the command format. The same matching applies when authorization is done using local profiles, RADIUS and TACACS+.

Table 4: Authorization and matches based on the command format

Profile entry format	Classic CLI	MD-CLI	NETCONF	gRPC/gNMI
Classic CLI	Yes	Maybe	Maybe	Maybe
MD-CLI	Maybe	Yes	Yes	Yes

Note: Nokia recommends testing the profiles in each CLI engine to confirm the desired action occurs.

2.1.2.5 Configuring authorization

The following section authorization describes how to configure authorization for some specific cases.

2.1.2.5.1 System-provisioned local authorization profiles

SR OS provides the following built-in (system-provisioned) local command authorization profiles. These profiles can be removed or modified.

- default
- administrative

The built-in profiles are applicable to users using the classic CLI or MD-CLI, and contain rules that apply to classic CLI and rules that apply to MD-CLI interfaces in the same profile.

By default, in SR OS, the administrative profile is associated with the default user named `admin`.

In MD-CLI, a newly-created user is not associated with any profile. The user can manually associate a user with the default profile if required.

In the classic CLI, the default profile is automatically assigned to any newly-created user, but the user can remove the profile from any user and replace it with another profile. The classic CLI also has an internal mechanism that denies access to **show system security** commands for all users, so users must be given access to these commands with a permit entry in a profile.

2.1.2.5.2 Authorization for configuration groups

Note: This information applies to the MD-CLI and to NETCONF.

To configure authorization for configuration groups explicitly, create an entry for the group configuration in the user's profile.

For example, to deny access to router interfaces in both the main configuration branch and in the group configuration branch, create an entry for each one.

In the following example, entry 10 prevents the user from viewing, creating, and editing router interfaces in the main configuration branch and from inheriting router interface configurations from configuration groups. Entry 20 prevents the user from viewing, creating, and editing router interfaces in the group configuration branch.

Example: MD-CLI

```
[ex:/configure system security aaa local-profiles profile "exampleProfile"]
A:admin@node-2# info
    entry 10 {
        match "configure router interface"
        action deny
    }
    entry 20 {
        match "configure groups group router interface"
        action deny
    }
```

2.1.2.5.3 Authorization for NETCONF operations

Authorization for YANG-modeled operations that are available over NETCONF can be configured for each operation in addition to authorization for the input and output paths that these operations access. See YANG-modeled operations in the *7705 SAR Gen 2 System Management Guide*.

Match YANG-modeled operations by specifying the path in command profile entries. Authorization is configured as follows, when:

- The CLI path is the same as the YANG-modeled operation path.

Use the CLI patch to configure matching. For example, the **oam**, **ping**, **pyexec**, and **traceroute** commands have the same paths.

Example: MD-CLI

```
[ex:/configure system security aaa local-profiles profile "netconf"]
A:admin@node-2# info
    entry 10 {
        match "ping"
        action permit
    }
```

Example: classic CLI

```
A:node-2>config>system>security>profile# info
----------------------------------------------
                entry 10
                    match "ping"
                    action permit
                exit
----------------------------------------------
```

- YANG-modeled actions have input leafs that are only available over NETCONF.

Configure matching by using the CLI path, including the additional input leafs. For example, the **perform system ssh generate-keypair** command has **passphrase** and **force** leafs that are only available over NETCONF.

Example: MD-CLI

```
[ex:/configure system security aaa local-profiles profile "netconf"]
A:admin@node-2# info
    entry 20 {
        match "perform system ssh generate-keypair passphrase"
        action permit
    }
```

Example: classic CLI

```
A:node-2>config>system>security>profile# info
----------------------------------------------
                entry 20
                    match "perform system ssh generate-keypair passphrase"
                    action permit
                exit
----------------------------------------------
```

- There is no CLI path for the YANG-modeled operation.

Matching is configured using the remote procedure call (RPC) name in the oper model. For example, the <md-compare> RPC is only available over NETCONF.

Example: MD-CLI

```
[ex:/configure system security aaa local-profiles profile "netconf"]
A:admin@node-2# info
    entry 30 {
        match "md-compare"
        action permit
    }
```

Example: classic CLI

```
A:node-2>config>system>security>profile# info
----------------------------------------------
                entry 30
                    match "md-compare"
                    action permit
                exit
----------------------------------------------
```

The following usage guidelines apply to authorization for NETCONF operations:

- These paths are not subject to authorization because they have read-only state information about the supported YANG models and system capabilities that are required by NETCONF controllers:
  - /modules-state
  - /netconf
  - /netconf-state
  - /yang-library

- The NETCONF **base-op-authorization action** user profile permissions have a higher priority than command profile entries.
  For example, when the following commands are disabled in the user profile, the user does not have access to any NETCONF actions even when an entry permitting one, for example **ping**, exists in the command profile. When the following commands are enabled in the user profile, then the matching entry in the command profile applies.
- **MD-CLI**

```
configure system security aaa local-profiles profile netconf base-op-authorization action
```

- **classic CLI**

```
configure system security profile netconf>base-op-authorization action
```

2.1.2.6 Authorizing delete operations in model-driven interfaces

Authorization of configuration delete operations in model-driven interfaces is configured by matching the **delete** parameter in local profiles, RADIUS VSAs, or TACACS+ VSAs. While the **delete** parameter is used for delete operations in the MD-CLI, NETCONF and gRPC/gNMI each use their respective RPCs. The MD-CLI command format is used for all model-driven interface authorization. See Authorization profiles for classic and model-driven management interfaces for more information.

Note: The **delete** parameter must be specified first in the matching command list.

The following examples deny the deleting BGP configuration, while still allowing the BGP configuration to be added or modified.

Example: Local profiles

The following example shows the MD-CLI configuration with local profiles.

```
[ex:/configure system security aaa local-profiles profile "restrictive"]
A:admin@node-2# info
     entry 10 {
        match "delete configure router bgp"
        action deny
    }
    entry 20 {
        match "configure router bgp"
        action permit
    }

[ex:/configure router "Base"]
A:admin@node-2# bgp router-id 10.10.10.10

*[ex:/configure router "Base"]
A:admin@node-2# delete bgp
MINOR: MGMT_CORE #2020: Permission denied - unauthorized use of 'bgp'
```

The following example shows the RADIUS server configuration.

Example: RADIUS server configuration

```
user1
```

```
    Timetra-Cmd = "delete configure router bgp"
    Timetra-Action = deny
    Timetra-Cmd = "configure router bgp"
    Timetra-Action = permit
```

The following example show the TACACS+ server configuration.

Example: TACACS+ server configuration

```
user = user1 {
    service = nokia-user-profile {
        entry10 = "delete configure router bgp"
        action10 = "deny"
        entry20 = "configure router bgp"
        action20 = "permit"
    }
}
```

2.1.2.6.1 TACACS+ interactive command authorization of delete operations in model-driven interfaces

In addition to authorizing the configuration delete operation using TACACS+ VSAs, the system can alternatively be configured to always send the delete operation in the **cmd** argument and the path in the **cmd-arg** argument in TACACS+ interactive authorization requests. All delete operations use the same **cmd=delete** request format in model-driven interfaces. Use the following commands to configure the system to send **cmd=delete** for delete operations.

```
configure system security aaa remote-servers tacplus authorization request-format access-
operation-cmd delete
configure service vprn aaa remote-servers tacplus authorization request-format access-
operation-cmd delete
```

Note: The delete operation can be used anywhere in MD-CLI input.

```
delete configure system name
configure delete system name
configure system delete name
```

Example: AV pairs sent for delete operation

The following example shows the AV pairs that are sent for a delete operation.

```
[ex:/configure system]
A:admin@node2# delete name

# Command authorization
cmd=delete
cmd-arg=configure
cmd-arg=system
cmd-arg=name
# Resulting change authorization
cmd=delete
cmd-arg=configure
cmd-arg=system
cmd-arg=name
```

Example: TACACS+ server configuration

The following example shows the TACACS+ server configuration to deny access to delete **configure system name** and to permit deleting all other commands.

```
user = admin {
    cmd = delete {
        deny "configure system name"
        permit .*
    }
}
```

2.1.3 Accounting

2.1.3.1 RADIUS accounting

Accounting can be configured independently from RADIUS authorization and RADIUS authentication for two purposes:

- CLI command accounting
- Enhanced Subscriber Management subscriber host accounting

When enabled, RADIUS accounting sends accounting requests from the router to the RADIUS server on UDP port 1813, or TCP port 2083 if TLS is configured. The server receives accounting requests and returns a response to the router indicating that it has successfully received the request. Each request issued on the router generates a record sent to the RADIUS server. If no response is received in the timeout period, the accounting request is retransmitted until the retry count is exhausted. In this case, the router sends the accounting request to the next configured RADIUS server.

User information, passwords and authentication keys are never transmitted as part of the accounting request.

2.1.3.2 TACACS+ accounting

The SR OS allows the administrator to configure the type of accounting record packet that is to be sent to the TACACS+ server when specified events occur on the device. The **accounting record-type** command option indicates whether TACACS+ accounting start and stop packets will be sent or just stop packets be sent. Start/stop messages are only sent for individual commands, not for the session.

The router checks the configuration to see if TACACS+ accounting is required for the particular event when:

- a user logs in to request access to the network using Telnet or SSH
- a user enters a command for which accounting command options are configured
- a system event occurs, such as a reboot or a configuration file reload

If TACACS+ accounting is required, then, depending on the accounting record type specified, the device sends a start packet to the TACACS+ accounting server that contains information about the event.

The TACACS+ accounting server acknowledges the start packet and records information about the event. When the event ends, the device sends a stop packet. The stop packet is acknowledged by the TACACS+ accounting server.

2.1.3.3 Command accounting log events

In addition to RADIUS and TACACS+ accounting, SR OS supports a set of log events dedicated to command accounting.

For the following log events related to command accounting, see the *7705 SAR Gen 2 Log Events Guide*:

- cli_user_io
- snmp_user_set
- cli_config_io
- cli_unauth_user_io
- cli_unauth_config_io
- md_cli_io
- md_cli_unauth_io
- netconf_auth
- netconf_unauth
- grpc_auth
- grpc_unauth

2.1.4 Server health check monitoring

Health check monitoring of the LDAP, RADIUS, and TACACS+ servers is performed 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.

When health check monitoring is disabled, the operational status for the server is up if a response to the last request was received.

Use the following command to configure health check monitoring:

- **MD-CLI**

```
configure system security aaa health-check
```

- **classic CLI**

```
configure system security password health-check
```

Different health check monitoring requests are sent depending on the server type.

- LDAP – A TCP connection to the server is opened and closed with an Unbind operation.
- 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.

The servers are accessed in order from lowest to highest specified index (from 1 to 5) for authentication requests until a response from a server is received. A higher indexed server is only queried if no response is received from a lower indexed server. If a response from the server is received, no other server is queried.

2.1.5 User access control method ordering

The user can configure routers to use RADIUS, TACACS+, LDAP, and local AAA methods to validate users requesting access to the network. The order in which requests are processed among RADIUS, TACACS +, LDAP, and local methods can be specifically configured. For example, the authentication order can be configured to process authorization using TACACS+ first, then RADIUS for authentication and accounting. Local access can be specified last in the authentication order if the RADIUS and TACACS+ servers are not operational.

The following table lists security methods capabilities.

Table 5: Security methods capabilities

Method	Authentication	Authorization	Accounting
Local	✓	✓	✓[1]
TACACS+	✓	✓	✓[1]
RADIUS	✓	✓	✓[1]
LDAP	✓	Not supported	Not supported

Authentication and authorization request flow

Use the commands in the following context to define the authentication and authorization order shown in Figure 5: Security flow.

- **MD-CLI**

```
configure system security user-params authentication-order
```

- **classic CLI**

```
configure system security password authentication-order
```

The order is determined by specifying the sequence in which an authentication or authorization method is attempted when the **exit-on-reject** command option is disabled.

Example: Authentication and authorization method flow

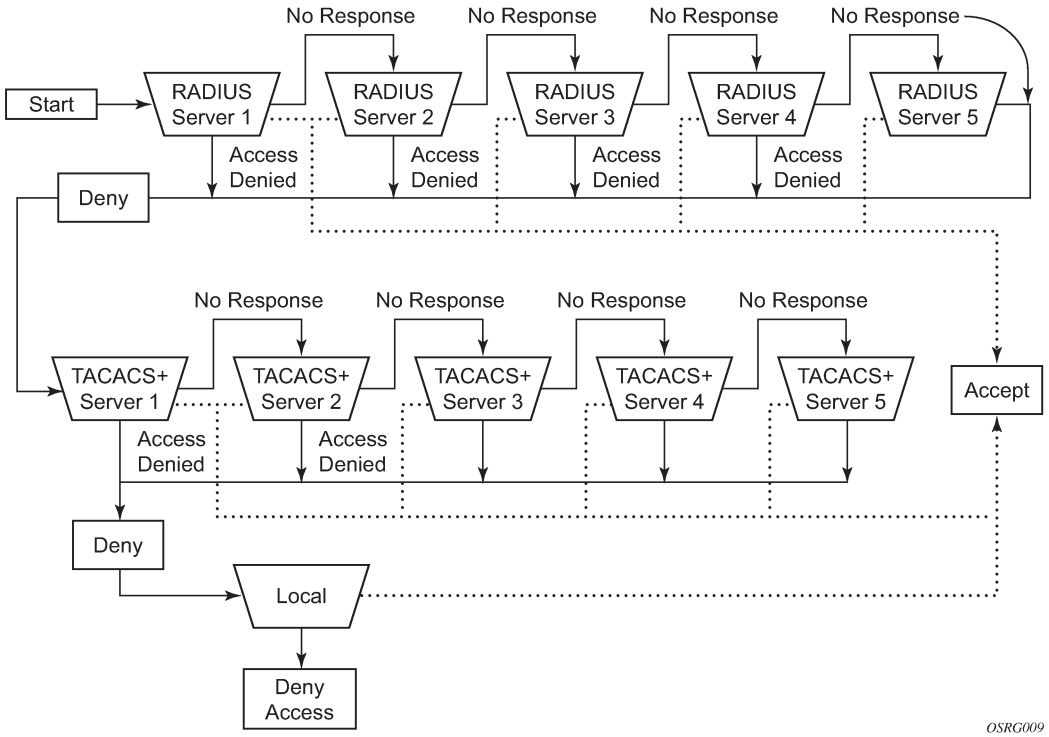
Figure 5: Security flow shows an example that uses the order of RADIUS, then TACACS+, and finally, local. A request is sent to RADIUS server 1. If there is no response from the server, the request is sent to the next RADIUS server and so on, until the last RADIUS server is attempted (RADIUS server 5). If server 5 does not respond, the request is sent to TACACS+ server 1. If there is no response from that server, the request is sent to the next TACACS+ server, and so on.

If a request is sent to an active RADIUS server and the username and password are not recognized, access is denied and the next method is attempted, in this case, the TACACS+ server. The process continues until the request is either permitted, denied, or each server is queried. Finally, if the request

[1] CLI commands are always logged in local command accounting log events such as cli_user_io or md_cli_io.

is denied by the active TACACS+ server, the local method is attempted. This is the last chance for the access request to be permitted.

Figure 5: Security flow



2.2 RADIUS VSAs

SR OS supports the configuration of Nokia-specific RADIUS attributes. These attributes are known as vendor-specific attributes (VSAs) and are defined in RFC 2138. If VSAs are not configured on the RADIUS server, the RADIUS user authenticates with options defined in the RADIUS default template when **use-default-template** is enabled. If VSAs are used, all mandatory VSAs must be configured for the RADIUS user to authenticate. It is up to the vendor to specify the format of their VSA. The attribute-specific field is dependent on the vendor definition of that attribute. The Nokia-defined attributes are encapsulated in a RADIUS vendor-specific attribute with the vendor ID field set to 6527, the vendor ID number. Nokia VSAs are defined in the `dictionary-freeradius.txt` file in the `support` folder of the software distribution.

The following RADIUS VSAs for AAA are supported:

- **Timetra-Access <ftp + console + netconf + grpc + scp-sftp + console-port-cli + ssh-cli + telnet-cli>**

  This VSA specifies the router management access methods a user can access. Multiple access methods can be specified by adding the value of the access methods to allow in the RADIUS server configuration file. This VSA is mandatory when the RADIUS default template is disabled. This VSA is mandatory when the RADIUS default template is disabled. It corresponds to the following commands:

– **MD-CLI**

```
configure system security user-params local-user user access
```

– **classic CLI**

```
configure system security user access
```

For example, to allow console port CLI, SSH CLI and NETCONF access:

```
Timetra-Access = 100 # 32 (console-port-cli) + 64 (ssh-cli) + 4 (netconf)
```

or:

```
Timetra-Access = console-port-cli
Timetra-Access += ssh-cli
Timetra-Access += netconf
```

- **Timetra-Profile <string>**

  This VSA is mapped to the user profile which must be configured on the router. It corresponds to the following commands:

  – **MD-CLI**

  ```
  configure system security user-params local-user user console member
  ```

  – **classic CLI**

  ```
  configure system security user console member
  ```

  The following guidelines apply to local and remote authentication:

  – The **authentication-order** configured on the router must include the **local** command option.

  – The username may or may not be configured on the router.

  – The user must be authenticated by the RADIUS server.

  – A temporary profile is created when the user authenticates that is evaluated before local profiles. The Timetra-Default-Action VSA must be sent with a value of **none** so that the local profiles are evaluated after the temporary profile if the RADIUS default template is disabled. The profiles and evaluation order assigned to the user can be displayed with the **show system security user detail** command.

  – Up to eight local profiles can exist on the router for a user. The sequence in which the profiles are specified is relevant. The most explicit matching criteria must be ordered first. The process stops when the first complete match is found.

  If all of the preceding conditions are not met, access to the router is denied and a failed login event or trap is written to the security log.

- **Timetra-Action <permit | deny | read-only>**

  This VSA specifies the action when the user has entered a command specified in the Timetra-Cmd VSA, and is mandatory in sequence after the Timetra-Cmd VSA. It corresponds to the following commands:

– **MD-CLI**

```
configure system security aaa local-profiles profile entry action
```

– **classic CLI**

```
configure system security profile entry action
```

- **Timetra-Default-Action <permit-all | deny-all | read-only-all | none>**

  This VSA specifies the default action when the user has entered a command, and no entry configured in the Timetra-Cmd VSA for the user has matched. This VSA is mandatory when the RADIUS default template is disabled. It corresponds to the following commands:

  – **MD-CLI**

  ```
  configure system security aaa local-profiles profile default-action
  ```

  – **classic CLI**

  ```
  configure system security profile default-action
  ```

- **Timetra-Cmd <string>**

  This VSA configures a command or command subtree as the scope for the match condition for the action specified in the Timetra-Action VSA, and is mandatory in sequence before the Timetra-Action VSA. The command and all commands in subtrees are authorized. If an invalid command is specified, the radiusUserProfileInvalid event is logged and authentication fails. It corresponds to the following commands:

  – **MD-CLI**

  ```
  configure system security aaa local-profiles profile entry match
  ```

  – **classic CLI**

  ```
  configure system security profile entry match
  ```

- **Timetra-Home-Directory <string>**

  This VSA specifies the home directory. It cannot be used to delete a home directory that is configured in the RADIUS default template. It corresponds to the following commands:

  – **MD-CLI**

  ```
  configure system security user-params local-user user home-directory
  ```

  – **classic CLI**

  ```
  configure system security user home-directory
  ```

- **Timetra-Restrict-To-Home <true | false>**

  This VSA specifies whether user access is limited to their home directory (and directories and files in the home directory). If this VSA is not configured, the user is allowed to access the entire file system. It corresponds to the following commands:

– **MD-CLI**

```
configure system security user-params local-user user restricted-to-home
```

– **classic CLI**

```
configure system security user restricted-to-home
```

- **Timetra-Save-When-Restricted <true | false>**

  When this VSA is set to true, the user can execute configuration save operations (for example, **admin save**) via any management interface (CLI, NETCONF, and so on) when Timetra-Restrict-To-Home is set to true. It corresponds to the following commands:

  – **MD-CLI**

  ```
  configure system security user-params local-user user save-when-restricted
  ```

  – **classic CLI**

  ```
  configure system security user save-when-restricted
  ```

- **Timetra-Exec-File <string>**

  This VSA specifies a user login exec file, which executes whenever the user successfully logs in to a console session. It corresponds to the following commands:

  – **MD-CLI**

  ```
  configure system security user-params local-user user console login-exec
  ```

  – **classic CLI**

  ```
  configure system security user console login-exec
  ```

- **Timetra-NETCONF-BaseOp <cancel-commit | close-session | commit | copy-config | create-subscription | delete-config | discard-changes | edit-config | get | get-config | get-data | get-schema | kill-session | lock | validate>**

  This VSA specifies the NETCONF operations that the user can execute. It corresponds to the following commands:

  – **MD-CLI**

  ```
  configure system security aaa local-profiles profile netconf base-op-authorization
  ```

  – **classic CLI**

  ```
  configure system security profile netconf base-op-authorization
  ```

  Multiple operation values can be specified in any order separated by semicolons (;) in the RADIUS server configuration file, for example.

  ```
  Timetra-NETCONF-BaseOp = close-session;commit;discard-changes;edit-config;get-
  config;lock;validate
  ```

Multiple RPCs can also be specified on different lines in the RADIUS server configuration file, for example.

```
Timetra-NETCONF-BaseOp = close-session,
Timetra-NETCONF-BaseOp += commit,
Timetra-NETCONF-BaseOp += discard-changes,
Timetra-NETCONF-BaseOp += edit-config,
Timetra-NETCONF-BaseOp += get-config,
Timetra-NETCONF-BaseOp += lock,
Timetra-NETCONF-BaseOp += validate
```

If an invalid operation is specified, the radiusUserProfileInvalid event is logged and authentication fails.

Note: The permissions for NETCONF RPCs are logically ORed from all profiles when multiple profiles are applied.

- **Timetra-NETCONF-Default-Action <permit | deny>**

  This VSA specifies the default action when the user has executed an operation, and no operation configured in the Timetra-NETCONF-BaseOp VSA for the user has matched. It is mandatory when the Timetra-NETCONF-BaseOp VSA is used. There is no corresponding configuration command.

- **Timetra-gRPC-RPC <gnmi-capabilities | gnmi-get | gnmi-set | gnmi-subscribe | gnoi-cert-mgmt-cangenerate | gnoi-cert-mgmt-getcert | gnoi-cert-mgmt-install | gnoi-cert-mgmt-revoke | gnoi-cert-mgmt-rotate | gnoi-file-get | gnoi-file-put | gnoi-file-remove | gnoi-file-stat | gnoi-file-transfertoremote | gnoi-system-cancelreboot | gnoi-system-ping | gnoi-system-reboot | gnoi-system-rebootstatus | gnoi-system-setpackage | gnoi-system-switchcontrolprocessor | gnoi-system-time | gnoi-system-traceroute | md-cli-session | rib-api-getversion | rib-api-modify>**

  This VSA specifies the gRPC RPCs the user can execute. It corresponds to the following commands:

  – **MD-CLI**

    ```
    configure system security aaa local-profiles profile grpc rpc-authorization
    ```

  – **classic CLI**

    ```
    configure system security profile grpc rpc-authorization
    ```

  Multiple RPCs can be specified in any order separated by semicolons (;) in the RADIUS server configuration file, for example.

  ```
  Timetra-gRPC-RPC = gnmi-capabilities;gnmi-subscribe
  ```

  Multiple RPCs can also be specified on different lines in the RADIUS server configuration file, for example.

  ```
  Timetra-gRPC-RPC = gnmi-capabilities,
  Timetra-gRPC-RPC = gnmi-subscribe
  ```

  If an invalid RPC is specified, the radiusUserProfileInvalid event is logged and authentication fails.

  Note: The permissions for gRPC RPCs are logically ORed from all profiles when multiple profiles are applied.

- **Timetra-gRPC-Default-Action <permit | deny>**

  This VSA specifies the default action when the user has executed an RPC, and no RPC configured in the Timetra-gRPC-RPC VSA for the user has matched. It is mandatory when the Timetra-gRPC-RPC VSA is used. This is no corresponding configuration command.

2.2.1 RADIUS configuration for file access control using VSAs

Configure file access control in one of the following ways depending on the file access requirements of users:

- locally with no VSAs
- with VSAs

Note: File access is denied when the **restricted-to-home** command is configured, unless the **home-directory** command is configured and the directory is created by an administrator.

Example: RADIUS server with VSA configuration for per-user home directories, and a locally configured default template for other options

This example shows the following configuration:

- All users can save the configuration.
- On the router: each user has a home directory with restricted file access. The administrator must create the home directory for each user.
- On the RADIUS server: the home directory is configured with a VSA.
- On the router: optionally, access to copy files to the router with SCP/SFTP is configured in the RADIUS default template.
- On the router: other file access controls are configured in the RADIUS default template.

RADIUS server configuration

```
user1
    Timetra-Home-Directory = "cf3:\users\user1"

user2
    Timetra-Home-Directory = "cf3:\users\user2"

user3
    Timetra-Home-Directory = "cf3:\users\user3"

user4
    Timetra-Home-Directory = "cf3:\users\user4"
```

MD-CLI

```
[ex:/configure system security aaa user-template radius-default]
A:admin@node-2# info
    restricted-to-home true
    save-when-restricted true
```

Classic CLI

```
A:node-2>config>system>security>user-template# info
```

```
-------------------------------------------
               restricted-to-home
               save-when-restricted
 -------------------------------------------
```

Example: RADIUS server with VSA configuration and per-user home directories

This example shows the following configuration:

- All file access controls are configured with VSAs, which is the most flexible option to grant different file access to each user.
- On the router: each user has a home directory with restricted file access. The administrator must create the home directory for each user.
- On the router: the RADIUS default template is not used for file access.
- On the RADIUS server: access to copy files to the router with SCP/SFTP is configured with VSAs.
- On the RADIUS server: the administrator can also restrict file access to the home directory of the user and allow users to save the configuration based on the VSA value.

The user1 profile has access to all files and user1 can save the configuration and copy files to the router with SCP/SFTP.

RADIUS server configuration

```
user1
    Timetra-Access = 112 # 16 + (scp-sftp) + 32 (console-port-cli) + 64 (ssh-cli)
    # Timetra-Home-Directory is not defined
    Timetra-Restrict-To-Home = false
    # Timetra-Save-When-Restricted is not defined
```

The user2 profile has home directory access and user2 can save the configuration and copy files to the router with SCP/SFTP.

RADIUS server configuration

```
user2
    Timetra-Access = 112 # 16 + (scp-sftp) + 32 (console-port-cli) + 64 (ssh-cli)
    Timetra-Home-Directory = "cf3:\users\user2",
    Timetra-Restrict-To-Home = true,
    Timetra-Save-When-Restricted = true
```

The user3 profile has home directory access but user3 cannot save the configuration or copy files to the router with SCP/SFTP.

RADIUS server configuration

```
user3
    Timetra-Access = 96 # 32 (console-port-cli) + 64 (ssh-cli)
    Timetra-Home-Directory = "cf3:\users\user3",
    Timetra-Restrict-To-Home = true,
    Timetra-Save-When-Restricted = false
```

The user4 profile has no file access and user4 cannot save the configuration.

RADIUS server configuration

```
user4
    Timetra-Access = 96 # 32 (console-port-cli) + 64 (ssh-cli)
    # Timetra-Home-Directory is not defined
```

```
    Timetra-Restrict-To-Home = true,
    Timetra-Save-When-Restricted = false
```

2.3 TACACS+ services and VSAs

SR OS supports several services with VSAs for user access control. Administrators can optionally configure the services and VSAs for each user on a TACACS+ server instead of configuring them locally on the router.

Services and VSAs are available for the following user access control parameters:

- The **nokia-user** service and Table 6: VSAs for user profiles that correspond to the following commands:
  - **MD-CLI**

    ```
    configure system security user-params local-user user console
    ```

  - **classic CLI**

    ```
    configure system security user console
    ```

- The **nokia-user-profile** service and Table 7: VSAs for command authorization that correspond to the following commands:
  - **MD-CLI**

    ```
    configure system security aaa local-profiles profile default-action
    configure system security aaa local-profiles profile entry
    ```

  - **classic CLI**

    ```
    configure system security profile default-action
    configure system security profile entry
    ```

- The **nokia-user** service and Table 8: VSAs for file access control that correspond to the following commands:
  - **MD-CLI**

    ```
    configure system security user-params local-user user
    ```

  - **classic CLI**

    ```
    configure system security user
    ```

- The **nokia-user** service and Table 9: VSAs for system access methods that correspond to the following commands:
  - **MD-CLI**

    ```
    configure system security user-params local-user user access
    ```

  - **classic CLI**

    ```
    configure system security user access
    ```

- The **nokia-netconf-base-op-authorization** service and Table 10: VSAs for NETCONF RPCs that correspond to the following commands:
  - **MD-CLI**

    ```
    configure system security aaa local-profiles profile netconf base-op-authorization
    ```

  - **classic CLI**

    ```
    configure system security profile netconf base-op-authorization
    ```

- The **nokia-grpc-rpc-authorization** service and Table 11: VSAs for gRPC RPCs that correspond to the following commands:
  - **MD-CLI**

    ```
    configure system security aaa local-profiles profile grpc rpc-authorization
    ```

  - **classic CLI**

    ```
    configure system security profile grpc rpc-authorization
    ```

When a user authenticates with TACACS+, the router:

- when enabled, requests the **nokia-user**, **nokia-netconf-base-op-authorization**, **nokia-grpc-rpc-authorization**, and **nokia-user-profile** services and VSAs from the server for authorization after authentication succeeds
- uses the values from VSAs when present, and when not present uses the following:
  - values from the TACACS+ default template when the TACACS+ default template is enabled
  - disabled or no values when the TACACS+ default template is disabled
- rejects invalid VSA values and authentication fails
- rejects unknown mandatory VSAs and authentication fails
- discards unknown optional VSAs and authentication succeeds

Note: The **nokia-user-profile default-action** VSA and at least one **nokia-user access** VSA must be received when the TACACS+ default template (**use-default-template**) is disabled.

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

1. **nokia-user-profile** VSAs
2. privilege-level mapping
3. TACACS+ default template, when enabled
4. **deny-all** default action for the **nokia-user**, **nokia-netconf-base-op-authorization**, and **nokia-grpc-rpc-authorization** VSAs when they are not received

Example: User profile when the TACACS+ default template is enabled

The following behavior occurs for the user profile when the TACACS+ default template is enabled:

- the **nokia-user-profile** VSA is received, and the **nokia-netconf-base-op-authorization** and **nokia-grpc-rpc-authorization** VSAs are not received

- the user profile is applied from step 1
- the NETCONF and gRPC permissions are applied from step 3

Example: User profile when the TACACS+ default template is disabled

The following behavior occurs for the user profile when the TACACS+ default template is disabled:

- the **nokia-user profile** VSA is received, and the **nokia-netconf-base-op-authorization** and **nokia-grpc-rpc-authorization** VSAs are not received
- the user profile is applied from step 1
- the NETCONF and gRPC permissions are applied from step 4

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

Table 6: VSAs for user profiles

Service name	VSA name	Description	Values
nokia-user	login-exec	File to execute when the user logs in	A string up to 200 characters
nokia-user	profile	User profile	A string up to 32 characters. The profile must be configured on the router.

Table 7: VSAs for command authorization

Service name	VSA name	Description	Values
nokia-user-profile	default-action	Default action for command authorization. This VSA is mandatory when the **nokia-user-profile** service is used.	deny-all – sets the default of the profile to deny access to all commands permit-all – sets the default of the profile to permit access to all commands none – sets the default of the profile to no-action. This option is useful to assign multiple profiles to a user. read-only-all – sets the default of the profile to allow read-only access to all commands
nokia-user-profile	entry*N*	The user profile entry number *N*. More than one entry can be	A string up to 255 characters

Service name	VSA name	Description	Values
		created with unique numbers. Exits when the first match is found and executes the actions according to the accompanying action VSA. Entries must be sequenced from most explicit to least explicit.	Example: `entry1 = "configure router bgp"`
nokia-user-profile	action*N*	The action number *N* associated with entry number *N*	deny – denies commands matching the entry permit – permits commands matching the entry read-only – permits commands matching the entry with read-only access Example: **action1 = deny**

The following command authorization usage guidelines apply:

- The range of action or entry numbers is 1 to 9999.
- Leading zeros before the action or entry number are invalid, for example **entry01**.
- The entries are sorted by their entry number and evaluated in numerical order, the same way as evaluation in local profiles.
- One entry and one corresponding action number must exist, but they can be in any order in the TACACS+ configuration file.
- A maximum of 126 entries and 126 actions can be created.
- Per-command authorization, which has precedence over VSAs, must be disabled if it was previously enabled:
  - **MD-CLI**

    ```
    delete configure system security aaa remote-servers tacplus authorization
    ```

  - **classic CLI**

    ```
    configure system security tacplus no authorization
    ```

Table 8: VSAs for file access control

Service name	VSA name	Description	Values
nokia-user	home-directory	Home directory for the user	A string up to 200 characters
nokia-user	restricted-to-home	Restrict file access to the home directory of the user	true – denies the user from accessing files outside their home directory false – permits the user to access all files on the system
nokia-user	save-when-restricted	Save configurations when the user is restricted to home	true – allows configuration save operations for all configuration regions, for example, bof, debug, configure, or li via any management interface such as, CLI and NETCONF even if **restricted-to-home** is enabled false – denies saving the configuration when **restricted-to-home** is enabled

Note: One access method VSA is mandatory when the TACACS+ default template is disabled.

Table 9: VSAs for system access methods

Service name	VSA name	Description	Values
nokia-user	console-access	Allow console port CLI, SCP/SFTP, SSH CLI, and Telnet CLI access	true – permits access false – denies access
nokia-user	console-port-cli-access	Allow console port CLI access	true – permits access false – denies access
nokia-user	ftp-access	Allow FTP access	true – permits access false – denies access
nokia-user	grpc-access	Allow gRPC access	true – permits access false – denies access

Service name	VSA name	Description	Values
nokia-user	netconf-access	Allow NETCONF access	true – permits access false – denies access
nokia-user	scp-sftp-access	Allow SCP/SFTP access	true – permits access false – denies access
nokia-user	ssh-cli-access	Allow SSH CLI access	true – permits access false – denies access
nokia-user	telnet-cli-access	Allow Telnet CLI access	true – permits access false – denies access

Table 10: VSAs for NETCONF RPCs

Service name	VSA name	Description	Values
nokia-netconf-base-op-authorization	default-action	Default action when a VSA for an RPC is not defined. This VSA is mandatory when the **nokia-netconf-base-op-authorization** service is used.	permit-all – permits all RPCs deny-all – denies all RPCs
nokia-netconf-base-op-authorization	action	Allow the NETCONF <action> RPC	true – permits the RPC false –denies the RPC
nokia-netconf-base-op-authorization	cancel-commit	Allow the NETCONF <cancel-commit> RPC	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	close-session	Allow the NETCONF <close-session> RPC	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	commit	Allow the NETCONF <commit> RPC	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	copy-config	Allow the NETCONF <copy-config> RPC	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	create-subscription	Allow the NETCONF <create-subscription> RPC	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	delete-config	Allow the NETCONF <delete-config> RPC	true – permits the RPC false – denies the RPC

Service name	VSA name	Description	Values
nokia-netconf-base-op-authorization	discard-changes	Allow the NETCONF <discard-changes> RPC	true – permits the RPC false –denies the RPC
nokia-netconf-base-op-authorization	edit-config	Allow the NETCONF <edit-config> RPC	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	get	Allow the NETCONF <get> RPC	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	get-config	Allow the NETCONF <get-config> RPC	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	get-data	Allow the NETCONF <get-data> RPC	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	get-schema	Allow the NETCONF <get-schema> RPC	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	kill-session	Allow the NETCONF <kill-session> RPC	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	lock	Allow the NETCONF <lock> and <unlock> RPCs	true – permits the RPC false – denies the RPC
nokia-netconf-base-op-authorization	validate	Allow the NETCONF <validate> RPC	true – permits the RPC false – denies the RPC

Note: The permissions for NETCONF RPCs are logically ORed from all profiles when multiple profiles are applied.

Table 11: VSAs for gRPC RPCs

Service name	VSA name	Description	Values
nokia-grpc-rpc-authorization	default-action	Default action when a VSA for an RPC is not defined. This VSA is mandatory when the **nokia-grpc-rpc-authorization** service is used.	permit-all – permits all RPCs deny-all – denies all RPCs
nokia-grpc-rpc-authorization	gnmi-capabilities	Allow the gNMI Capabilities RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnmi-get	Allow the gNMI Get RPC	true – permits the RPC

Service name	VSA name	Description	Values
			false – denies the RPC
nokia-grpc-rpc-authorization	gnmi-set	Allow the gNMI Set RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnmi-subscribe	Allow the gNMI Subscribe RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-cert-mgmt-rotate	Allow the gNOI Certificate Rotate RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-cert-mgmt-install	Allow the gNOI Certificate Install RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-cert-mgmt-getcert	Allow the gNOI Certificate Get Certificates RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-cert-mgmt-revoke	Allow the gNOI Certificate Revoke Certificates RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-cert-mgmt-cangenerate	Allow the gNOI Certificate Can GenerateCSR RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-file-get	Allow the gNOI File Get RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-file-transfertoremote	Allow the gNOI File TransferToRemote RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-file-put	Allow the gNOI File Put RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-file-stat	Allow the gNOI File Stat RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-file-remove	Allow the gNOI File Remove RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-system-ping	Allow the gNOI System Ping RPC	true – permits the RPC false –denies the RPC
nokia-grpc-rpc-authorization	gnoi-system-traceroute	Allow the gNOI System Traceroute RPC	true – permits the RPC false – denies the RPC

Service name	VSA name	Description	Values
nokia-grpc-rpc-authorization	gnoi-system-time	Allow the gNOI System Time RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-system-setpackage	Allow the gNOI System SetPackage RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-system-switchcontrolprocessor	Allow the gNOI System SwitchControlProcessor RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-system-reboot	Allow the gNOI System Reboot RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-system-rebootstatus	Allow the gNOI System RebootStatus RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	gnoi-system-cancelreboot	Allow the gNOI System CancelReboot RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	md-cli-session	Allow the gNOI MdCli Session RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	rib-api-getversion	Allow the RibApi Get Version RPC	true – permits the RPC false – denies the RPC
nokia-grpc-rpc-authorization	rib-api-modify	Allow the RibApi Modify RPC	true – permits the RPC false – denies the RPC

Note: The permissions for gRPC RPCs are logically ORed from all profiles when multiple profiles are applied.

2.3.1 TACACS+ configuration for command authorization using VSAs

Command authorization can be configured with TACACS+ VSAs instead of with local profiles 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 profile is sent in VSAs from the TACACS+ server to the router after the user authenticates, and is then installed locally on the router in a temporary profile for each user. Using VSAs for command authorization also significantly reduces the latency with per-command authorization between the router and the TACACS+ server.

Example: TACACS+ server with VSAs for command authorization

This example shows the following configurations:

- On the router: all users are authenticated via TACACS+ and **use-default-template** is enabled. The TACACS+ default template can optionally be disabled if all user access and profile information is configured as VSAs on the TACACS+ server.

- On the router: the **nokia-user** and **nokia-user-profile** services are enabled.
- On the TACACS+ server: command authorization is configured.

Example: MD-CLI

```
[ex:/configure system security aaa remote-servers tacplus]
A:admin@node-2# info
    use-default-template true
    service-request {
        nokia-user true
        nokia-user-profile true
    }
```

Example: classic CLI

```
A:node-2>config>system>security>tacplus# info
----------------------------------------------
                use-default-template
                service-request
                    nokia-user
                    nokia-user-profile
                exit
----------------------------------------------
```

Example

The user1 profile uses the **profile** VSA to assign the user the `noc-staff` profile, which is configured locally on the router.

TACACS+ server configuration

```
user = user1 {
    service = nokia-user {
        profile = "noc-staff"
    }
}
```

The user2 profile uses the **default-action**, **action** and **entry** VSAs to assign the user a per-command profile, which is installed as a temporary profile on the router.

TACACS+ server configuration

```
user = user2 {
    service = nokia-user-profile {
        default-action = deny-all
        action10 = "permit"
        entry10 = "show version"
        action20 = "permit"
        entry20 = "ping"
        action30 = "permit"
        entry30 = "logout"
    }
}
```

2.3.2 TACACS+ configuration for file access control using VSAs

Configure file access control in one of the following ways depending on the file access requirements of users:

- locally with no VSAs
- locally using the TACACS+ default template and some VSAs that are different for each user
- using the file access VSAs to control file access, and the TACACS+ default template for other user access controls

Note: File access is denied when the **restricted-to-home** command is configured unless the **home-directory** command is configured and the directory is created by an administrator.

Note: Some TACACS+ servers require the backslash character (\) to escape the backslash (\) character in quoted strings in the server configuration file (`tac_plus.conf`); for example:

- **home-directory = cf3:\users\user1**
- **home-directory = "cf3:\\users\\user1"**

Example: TACACS+ server with VSA configuration for per-user home directories, and a locally configured default template for other options

This example shows the following configurations:

- All users can save the configuration.
- On the router: each user has a home directory with restricted file access. The administrator must create the home directory for each user.
- On the router: the **nokia-user** service is enabled with the following commands:
  - **MD-CLI**

    ```
    configure system security aaa remote-servers tacplus service-request nokia-user
    ```

  - **classic CLI**

    ```
    configure system security tacplus service-request nokia-user
    ```

- On the TACACS+ server: the home directory is configured with a VSA.
- On the router: **use-default-template** is enabled, and optionally, access to copy files to the router with SCP/SFTP is configured in the TACACS+ default template.
- On the router: other file access controls are configured in the TACACS+ default template.

TACACS+ server configuration

```
user = user1 {
    service = nokia-user {
        home-directory = cf3:\users\user1
    }
}

user = user2 {
    service = nokia-user {
        home-directory = cf3:\users\user2
```

```
    }
}

user = user3 {
    service = nokia-user {
        home-directory = cf3:\users\user3
    }
}

user = user4 {
    service = nokia-user {
        home-directory = cf3:\users\user4
    }
}
```

Example: MD-CLI

```
[ex:/configure system security aaa user-template tacplus-default]
A:admin@node-2# info
    restricted-to-home true
    save-when-restricted true
```

Example: classic CLI

```
A:node-2>config>system>security>user-template# info
----------------------------------------------
                restricted-to-home
                save-when-restricted
 ----------------------------------------------
```

Example: TACACS+ server with VSA configuration and per-user home directories

This example shows the following configurations:

- All file access is controlled with VSAs, which is the most flexible option to grant different file access to each user.
- On the router: each user has a home directory with restricted file access. The administrator must create the home directory for each user.
- On the router: the **nokia-user** service is enabled with the following commands:
  - **MD-CLI**

    ```
    configure system security aaa remote-servers tacplus service-request nokia-user
    ```

  - **classic CLI**

    ```
    configure system security tacplus service-request nokia-user
    ```

- On the router: **use-default-template** is enabled and the TACACS+ default template is not used for file access.
- On the TACACS+ server: access to copy files to the router with SCP/SFTP is configured with VSAs.
- On the TACACS+ server: the administrator can also restrict file access to the home directory of the user and allow users to save the configuration based on the VSA value.

The user1 profile has access to all files and user1 can save the configuration, and copy files to the router with SCP/SFTP.

TACACS+ server configuration

```
user = user1 {
    service = nokia-user {
        scp-sftp-access = true
        # home-directory is not defined
        restricted-to-home = false
        # save-when-restricted is not defined
    }
}
```

The user2 profile has home directory access and user2 can save the configuration, and copy files to the router with SCP/SFTP.

TACACS+ server configuration

```
user = user2 {
    service = nokia-user {
        scp-sftp-access = true
        home-directory = cf3:\users\user2
        restricted-to-home = true
        save-when-restricted = true
    }
}
```

The user3 profile has home directory access but user3 cannot save the configuration or copy files to the router with SCP/SFTP.

TACACS+ server configuration

```
user = user3 {
    service = nokia-user {
        # scp-sftp-access is not defined
        home-directory = cf3:\users\user3
        restricted-to-home = true
        save-when-restricted = false
    }
}
```

The user4 profile has no file access and user4 cannot save the configuration or copy files to the router with SCP/SFTP.

TACACS+ server configuration

```
user = user4 {
    service = nokia-user {
        # scp-sftp-access is not defined
        # home-directory is not defined
        restricted-to-home = true
        save-when-restricted = false
    }
}
```

2.4 Control and management traffic protection

SR OS routers support an extensive set of configurable mechanisms to protect the CPU from being flooded with control or management traffic.

These protection mechanisms are a set of configurable hardware-based filters, classification, queuing, and rate-limiting functions that drop unwanted traffic before it reaches the control processor.

- in-band traffic extracted from the line cards to the CPM:
  - line card features:
    - ACLs filters: IPv4, IPv6, and MAC
    - anti-spoofing, uRPF
    - distributed CPU protection
  - CPM features:
    - CPM Filters: IPv4, IPv6, and MAC
    - centralized CPU Protection
    - per-peer queues, protocol queues, CPM queues
- out-of-band and in-band traffic: Management access filters

2.4.1 Centralized CPU protection

SR OS CPU protection is a centralized rate-limiting function that operates on the CPM to limit traffic destined for the CPU. The term "centralized CPU protection" is referred to as "CPU protection" in this guide and in the CLI to differentiate it from "Distributed CPU Protection".

CPU protection provides interface isolation by rate limiting the total amount of traffic extracted to the CPM per port, interface, or SAP in hardware using a combination of limits configurable at the CPU protection system level or as CPU protection policies assigned to access or network interfaces.

The following limits are configurable at the CPU protection system level:

- **link-specific rate**

  This applies to the link-specific protocols LACP (Ethernet LAG control) and Ethernet LMI (ELMI). The rate is a per-link limit (each link in the system has LACP/LMI packets limited to this rate).

- **port overall rate**

  This applies to all control traffic, the rate is a per-port limit, and each port in the system has control traffic destined for the CPM limited to this rate.

- **protocol protection**

  This blocks network control traffic for unconfigured protocols.

The following limits are configurable independently for access or network interfaces using a dedicated CPU protection policy:

- **overall rate**

  This applies to all control traffic destined for the CPM (all sources) received on an interface where the policy is applied. This is a per-interface limit. Control traffic received above this rate is discarded.

- **per-source rate**

  This is used to limit the control traffic destined for the CPM from each individual source. This per-source rate is only applied when an object (SAP) is configured with a CPU protection policy and also with the optional **mac-monitoring** or **ip-src-monitoring** commands. A source is defined as a *SAP, Source MAC Address* tuple for MAC monitoring and as a *SAP, Source IP Address* tuple for IP source monitoring.

Only specific protocols (as configured under **included-protocols** in the CPU protection policy) are limited (per source) when **ip-src-monitoring** is used.

- **out-profile rate**

  This applies to all control traffic destined for the CPM (all sources) received on an interface where the policy is applied. This is a per-interface limit. Control traffic received above this rate is marked as discard eligible (such as, out-profile/low-priority/yellow) and is more likely to be discarded if there is contention for CPU resources.

There are two default CPU protection policies for access and network interfaces.

Policy 254:

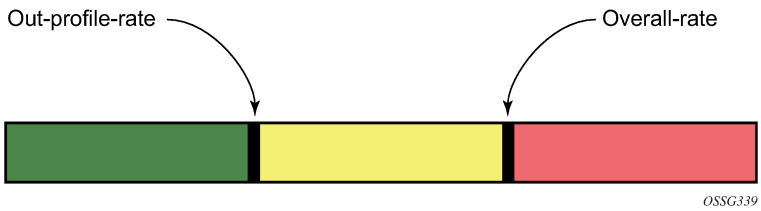
- This is the default policy that is automatically applied to access interfaces
- Traffic above 6000 pps is discarded
- overall-rate = 6000
- per-source-rate = max
- out-profile-rate = 6000

Policy 255:

- This is the default policy that is automatically applied to network interfaces
- Traffic above 3000 pps is marked as discard eligible, but is not discarded unless there is congestion in the queuing toward the CPU
- overall-rate = max
- per-source-rate = max
- out-profile-rate = 3000

A three-color marking mechanism uses a green, yellow, and red marking function. This allows greater flexibility in how traffic limits are implemented. The **out-profile-rate** command within the CPU protection policy maps to the boundary between the green (accept) and yellow (mark as discard eligible/low priority) regions. The **overall-rate** command marks the boundary between the yellow and red (drop) regions point for the associated policy (Figure 6: Profile marking).

Figure 6: Profile marking



If the overall rate is set to 1000 pps and as long as the total traffic that is destined for the CPM and intended to be processed by the CPU is less than or equal to 1000 pps, all traffic is processed. If the rate exceeds 1000 pps, protocol traffic is discarded (or marked as discard eligible/low priority in the case of the **out-profile-rate** command) and traffic on the interface is affected.

This rate limit protects all the other interfaces and ensures that a violation from one interface does not affect the rest of the system.

2.4.1.1 Protocol protection

Protocol protection allows traffic to be discarded for protocols not configured on the router. This helps mitigate DoS attacks by filtering invalid control traffic before it reaches the CPU. This is a feature of CPU Protection and can be enabled or disabled for the entire system.

When using the **protocol-protection** command, the system automatically maintains a per-interface list of configured protocols. For example, if an interface does not have IS-IS configured, then protocol protection discards any IS-IS packets received on that interface. Other protocols, such as L2TP, are controlled by the protocol protection at the VPRN service level.

Protocols controlled by the **protocol-protection** mechanism include:

- GTP
- IGMP
- IS-IS
- MLD
- L2TP control
- OSPFv2
- OSPFv3
- PPPoE
- PIM
- RIP
- PFCP

The following protocols are protected independently from protocol protection:

- The **per-peer-queuing** command protects BGP, LDP, T-LDP, MSDP, Telnet, and SSH.
- BFD control packets are dropped if BFD is not configured on a specific interface.

2.4.1.2 CPU protection extensions for ETH-CFM

CPU protection supports the ability to explicitly limit the amount of ETH-CFM traffic that arrives at the CPU for processing. ETH-CFM packets that are redirected to the CPU by either a Management Endpoint (MEP) or a Management Intermediate Point (MIP) will be subject to the configured limit of the associated policy. Up to four CPU protection policies may include up to ten individual ETH-CFM-specific entries. The ETH-CFM entries allow the operator to apply a packet-per-second rate limit to the matching combination of level and opcode for ETH-CFM packet that are redirected to the CPU. Any ETH-CFM traffic that is redirected to the CPU by a Management Point (MP) that does not match any entries of the applied policy is still subject to the overall rate limit of the policy itself. Any ETH-CFM packets that are not redirected to the CPU are not subject to this function and are treated as transit data, subject to the applicable QoS policy.

The operator first creates a CPU policy and includes the required ETH-CFM entries. Overlap is allowed for the entries within a policy, first match logic is applied. This means ordering the entries in the correct sequence is important to ensure the correct behavior is achieved. Even though the number of ETH-CFM entries is limited to ten, the entry numbers have a valid range from 1 to 100 to allow for ample space to insert policies between one and other.

Ranges are allowed when configuring the level and the OpCode. Ranges provide the operator a simplified method for configuring multiple combinations. When more than one level or OpCode is configured in this manner the configured rate limit is applied separately to each combination of level and OpCode match criteria. For example, if the levels are configured as listed in Table 12: Ranges versus levels and OpCodes, with a range of five (5) to seven (7) and the OpCode is configured for 3,5 with a rate of 1. That restricts all possible combinations on that single entry to a rate of 1 packet-per-second. In this example, six different match conditions are created.

Table 12: Ranges versus levels and OpCodes

Level	OpCode	Rate
5	3	1
5	5	1
6	3	1
6	5	1
7	3	1
7	5	1

When the policy is created, it must be applied to a SAP or binding within a service for these rates to take effect. This means the rate is on a per-SAP or per-binding basis. Only one policy may be applied to each SAP or binding. The **eth-cfm-monitoring** command must be configured in order for the ETH-CFM entries to be applied when the policy is applied to the SAP or binding. If this command is not configured, ETH-CFM entries in the policy are ignored. It is also possible to apply a policy to a SAP or binding by configuring the **eth-cfm-monitoring** command which does not have an MP. In this case, although these entries are enforced, no packets are redirected to the CPU.

By default, rates are applied on a per-peer basis. This means each individual peer is subject to the rate. Use the **aggregate** command to apply the rate to all peers. MIPs, for example, only respond to loopback messages and linktrace messages. These are typically on-demand functions and per-peer rate limiting is not required, making the aggregate function more appealing.

The **eth-cfm-monitoring** and **mac-monitoring** commands are mutually exclusive and cannot be configured on the same SAP or binding. The **mac-monitoring** command is used in combination with the traditional CPU protection and is not specific to ETH-CFM rate limiting feature described here.

When an MP is configured on a SAP or binding within a service which allows an external source to communicate with that MP, for example a User to Network Interface (UNI), **eth-cfm-monitoring** command with the **aggregate** command should be configured on all SAPs or bindings to provide the highest level of rate control.

The following example shows a policy configuration and the application of that policy to a SAP in a VPLS service configured with an MP. Policy 1 entry 10 limits all ETH-CFM traffic redirected to the CPU for all possible combinations to 1 packet-per-second. Policy 1 entry 20 limits all possible combinations to a rate of zero, dropping all request which match any combination. If entry 20 did not exist then only rate limiting of the entry 10 matches would occur and any other ETH-CFM packets redirected to the CPU would not be bound by a CPU protection rate.

Example: MD-CLI

```
[ex:/configure system security cpu-protection policy 1 eth-cfm]
A:admin@node-2# info
    entry 10 {
        pir 1
        level start 5 end 7 { }
        opcode start 3 end 3 { }
        opcode start 5 end 5 { }
    }
    entry 20 {
        pir 0
        level start 0 end 7 { }
        opcode start 0 end 255 { }
    }

[ex:/configure service vpls "10"]
A:admin@node-2# info
    sap 1/1/4:100 {
        admin-state enable
        cpu-protection {
            policy-id 1
            eth-cfm-monitoring {
                aggregate
            }
        }
        eth-cfm {
            mip primary-vlan none {
            }
        }
    }
```

Example: classic CLI

```
A:node-2>config>sys>security>cpu-protection#
  policy 1
    eth-cfm
      entry 10 level 5-7 opcode 3,5 rate 1
      entry 20 level 0-7 opcode 0-255 rate 0

A:node-2>config>service>vpls#
  sap 1/1/4:100
    cpu-protection 1 eth-cfm-monitoring aggregate
    eth-cfm
      mip
    no shutdown
```

2.4.2 TTL security

The SR OS TTL security evaluates the value of the incoming packets against a maximum TTL value configured in the system. This capability, also known as Generalized TTL Security Mechanism (GTSM) defined in RFC 5082, is supported for BGP, LDP, SSH and Telnet. If the incoming TTL value is less than the configured TTL value, the packets are discarded and a log is generated preventing attackers generating spoof traffic with larger number of hops than expected.

The TTL value is configurable on a per-peer basis for BGP and LDP and configurable at the system level for SSH and Telnet.

The TTL security mechanism was originally designed to protect the BGP infrastructure where the vast majority of ISP External Border Gateway Protocol (EBGP) peerings are established between adjacent routers. Because TTL spoofing cannot be performed, a mechanism based on an expected TTL value provides a simple and robust defense from infrastructure attacks based on forged BGP packets.

While TTL security is most effective in protecting directly-connected BGP or LDP peers, it can also provide protection to multihop sessions. For multihop sessions the expected TTL value can be set to 255 minus the configured range of hops.

2.4.3 Management Access Filter

The CPM CPU uses Management Access Filter (MAF) filters to perform filtering that applies to both traffic from the line cards directed to the CPM CPU as well as traffic from the management Ethernet port.

2.4.3.1 MAF filter packet match

You can configure three different management-access filter policies: IP filter, IPv6 filter, and MAC filter. Each policy is an ordered list of entries. For this reason, you must sequence the entries correctly from the most to the least explicit.

Management Access filter (MAF) packet match rules:

- Each MAF policy is an ordered list of entries, therefore entries must be sequenced correctly from the most to the least explicit.
- If multiple match criteria are specified in a single MAF filter policy entry, all criteria must be met for the packet to be considered a match against that policy entry (logical AND).
- Any match criteria not explicitly defined is ignored during a match.
- A MAF filter policy entry with match criteria defined, but no action configured, inherits the default action.
- The default action for the **management-access-filter** filter entry applies individually per IPv4, IPv6, or MAC CPM filter policies that are in an enabled state.
- When both **mac-filter** and **ip-filter** or **ipv6-filter** are applied to a specific packet, the **mac-filter** is applied first.

2.4.3.2 MAF IPv4/IPv6 filter entry match criteria

The following table lists the supported IPv4 and IPv6 match criteria.

Table 13: IPv4 and IPv6 match criteria

Criteria	Description
src-ip	Matches the specified source IPv4 or IPv6 address prefix and mask against the source IPv4 or IPv6 address field in the IP packet header. IPv4 and IPv6 matching prefix-lists can be used to enhance matching capabilities.
next-header	Matches the specified upper-layer protocol (such as TCP, UDP, or IGMPv6) against the next-header field of the IPv6 packet header. "*" can be used to specify a TCP or UDP upper-layer protocol

Criteria	Description
	match (Logical OR). Next-header matching allows also matching on presence of a subset of IPv6 extension headers. See the *7705 SAR Gen 2 Classic CLI Command Reference Guide* for more information about which extension header match is supported.
protocol	Matches the specified protocol against the Protocol field in the IPv4 packet header (for example, TCP, UDP, or IGMP) of the outer IPv4. "*" can be used to specify TCP or UDP upper-layer protocol match (Logical OR).
dst-port	Matches the specified port value against the destination port number of the UDP or TCP packet header.
flow-label	Matches the IPv6 flow label.
router	Matches the router instance that packets that are ingressing from for this filter entry.
src-port	Matches the port that packets are ingressing from for this filter entry.

2.4.3.3 MAF MAC filter entry match criteria

Table 14: Router instance match criteria describes the supported MAC match criteria. The criteria are evaluated against the Ethernet header of the Ethernet frame.

Table 14: Router instance match criteria

Criteria	Description
frame-type	Matches a specific type of frame format.
src-mac	Matches the specified source MAC address frames. Optionally, operators can configure a mask to be used in a match.
dst-mac	Matches the specified destination MAC address frames. Optionally, operators can configure a mask to be used in a match.
dot1p	Matches 802.1p frames. Optionally, operators can configure a mask to be used in a match.
etype	Matches the specified Ethernet II frames. The Ethernet type field is a two-byte field used to identify the protocol carried by the Ethernet frame.
snap-oui	Matches frames with the specified three-byte OUI field.
snap-pid	Matches frames with the specified two-byte protocol ID that follows the three-byte OUI field.

Criteria	Description
ssap	Matches the specified frames with a source access point on the network node designated in the source field of the packet. Optionally, operators can configure a mask to be used in a match.
dsap	Matches the specified frames with a destination access point on the network node designated in the destination field of the packet. Optionally, operators can configure a mask to be used in a match.
CFM opcode	Matches the specified packet with the specified CFM opcode.
service ID	Matches the service ID packets are ingressing from.
service name	Matches the service name packets are ingressing from.

2.4.3.4 MAF filter policy action

Management-access filter policies support the following traffic configuration actions:

- **MD-CLI**
  Supported actions are **ignore-match**, **accept**, **drop**, and **reject**.
- **classic CLI**
  Supported actions are **permit**, **deny**, and **deny-host-unreachable**.

2.4.3.5 MAF filter policy statistics and logging

Management access filter match count can be displayed using **show** commands. Logging is recorded in the system security logs.

2.5 Other security features

This section describes other security features supported by the SR OS.

2.5.1 SSH

The Secure Shell (SSH) protocol provides a secure, encrypted Telnet-like connection to a router. A connection is always initiated by the client (the user). Authentication uses one of the configured authentication methods (local, RADIUS, TACACS+, or LDAP). With authentication and encryption, SSH allows for a secure connection over an insecure network.

SR OS supports SSH version 2 (SSHv2) only. When a configuration contains SSHv1, SSHv1 is deprecated from the configuration, and the configuration migrates to SSHv2 using the default cipher list.

SSH runs on top of a transport layer (like TCP or IP) and provides authentication and encryption capabilities.

SR OS has a global SSH server process to support inbound SSH, SFTP, NETCONF, and SCP sessions initiated by external client applications. This server process is separate from the SSH and SCP client commands on the routers which initiate outbound SSH and SCP sessions.

Inbound SSH, Telnet, and FTP sessions are counted separately. Use the following command to set the limit for each type separately.

```
configure system login-control
```

However, there is a maximum total of 50 sessions for SSH and Telnet together. SCP, SFTP, and NETCONF sessions are counted as SSH sessions.

When the SSH server is enabled, an SSH security key is generated. Unless the **preserve-key** command option is configured for SSH, the security key is only valid until the node is restarted or the SSH server is stopped and restarted. For RSA, the key size is non-configurable and set to 2048 for SSHv2 RSA. When the server is enabled, all inbound SSH, SCP, SFTP, and NETCONF sessions are accepted provided the session is properly authenticated.

When the global SSH server process is disabled, no inbound SSH, SCP, SFTP, or NETCONF sessions are accepted.

When using SCP to copy files from an external device to the file system, the SCP server accepts either forward slash ("/") or backslash ("\") characters to delimit directory and filenames. Similarly, the SCP client application can use either slash or backslash characters, but not all SCP clients treat backslash characters as equivalent to slash characters. In particular, UNIX systems can interpret the backslash character as an "escape" character which is not transmitted to the SCP server. For example, a destination directory specified as "`cf1:\dir1\file1`" is transmitted to the SCP server as "`cf1:dir1file1`", where the backslash escape characters are stripped by the SCP client system before transmission. On systems where the client treats the backslash like an "escape" character, a double backslash "\\" or the forward slash "/" can be used to properly delimit directories and the filename.

There are three pairs of configurable lists: cipher lists, MAC lists, and KEX lists. In each pair, one list is dedicated to the SSH server, and a second list is dedicated to the SSH/SCP client. These can be configured for negotiation of the best compatible cipher, MAC, and KEX algorithm between the client and server. The lists can be created and managed under the **security ssh** context. The client lists are used when the SR OS is acting as the SSH client and the server lists are used when the SR OS is acting as a server. The first algorithm matched on the lists between the client and server is the preferred algorithm for the session.

SR OS supports the following SSHv2 authentication methods:

- password
- keyboard-interactive
- public key

SR OS SSH supports multichannel within a single connection. The primary connection authenticates the user through PKI or keyboard authentication. After the primary connection is authenticated, applications like NETCONF can open multiple channel "sessions" to the server with the same connection. SR OS supports one application with multiple channels over the same connection. Each connection can have 50 channels for a maximum of 50 channels per system.

2.5.1.1 SSH and Telnet ports configurable to non-default port value

For security reasons, the SSH and Telnet ports are configurable to a non-default port value. This makes it harder for unauthorized users to scan these ports and launch Denial of Service (DoS) attacks. On SR OS SSH and Telnet servers, the listening SSH and, respectively, Telnet ports are configurable. In addition, the SR OS SSH and Telnet clients can initiate an SSH or Telnet connection to a specific non-default SSH or Telnet port.

SSH or Telnet listening port configuration is not supported for the VPRN Global Routing Table (GRT) leaking (allow management) feature. Consequently, management via VPRN (GRT leaking) is disabled as soon as the port is changed to non-default port.

2.5.1.1.1 Application configuration for SSH and Telnet

The listening port value can be configured in the range of 1024 to 49151. The SSH and Telnet ports should not be configured in the respective well-known port ranges.

2.5.1.1.2 Ensuring no other application is using the SSH or Telnet port

When the user attempts to configure a new value for the default SSH or Telnet port, the SSH or Telnet application, respectively, checks whether the port is in use by another protocol. If the port is in use, the configuration is blocked and a warning is displayed that "Port is already in use.".

The newly configured port cannot be overwritten by any other protocol even if there is no active SSH or Telnet session. After the port is configured, no other protocol is allowed to reserve the port.

2.5.1.1.3 Behavior in base routing and VPRN

In a VPRN, after the SSH or Telnet port is changed, the port used previously is closed via the SR OS firewall and all sessions on that port are dropped.

In base routing, after the SSH or Telnet port is changed, the previous port remains open and existing sessions are not disconnected, but new connections cannot be established on that port.

Note: For base routing, Nokia recommends that CPM filters be set to block unwanted ports and IP addresses.

2.5.1.1.4 NETCONF behavior

The following system behaviors apply:

- There is only one instance of an SSH server in the system, but it can bind to and listen on multiple TCP ports: one for NETCONF and the second one for other SSH services.
- The port for NETCONF and the port for other SSH services are independent of each other.
- The SSH server (for other services) can be configured to listen on ports 22 and 1024 to 49151.
- NETCONF can be configured to listen on ports 830 or 22 only and does not follow the SSH port change.

2.5.1.1.5 Known limitations

If there are open SSH or Telnet connections to the previously configured port via VPRN, the attempt to change the port back to the previous port is blocked for approximately 5 minutes. This is caused by the firewall blocking the open connections from being fully closed. For example:

1. SSH is connected to port 22 by default. If the user changes the port to 22000, the connections to port 22 are immediately terminated.
2. Immediately after the first port change, if the user tries to change the port back to 22, the system returns an error and the port is not changed. The user must wait for approximately 5 minutes after the initial port change to 22000 to change the port back to the previously configured port. This limitation applies only if reverting from the current port to the previously configured port. There is no limitation if the user changes to a new port; for example, if the user changes the port to 22022.

When SSH or Telnet connections to the initial port exist via inband or OOB, the port change back to the initial value is blocked until all connections to the initial port are closed. For example:

- If the port is changed and there are open connections to the changed port via inband or OOB, the connections must be closed manually to be able to revert to the previous port.
- If SSH is connected to port 22 and the user changes the port to 22000, the change is effective immediately, but the SSH connection to port 22 remains open.
- Until all connections to port 22 are closed, the port cannot be changed back from 22000 to 22.

2.5.1.1.6 Distributed CPU aspects

DCPU rate limiting is supported for newly configured SSH ports as follows:

- when SSH is moved from the default port to a new port
- when SSH is moved back from the non-default configured port to the default port

2.5.1.2 SSH PKI authentication

The SSH server can support a public key authentication (also known as Public Key Infrastructure (PKI)), if the server was previously configured to know the client's public key.

Using PKI can be more secure than the existing username and password method for the following reasons.

- A user will typically reuse the same password with multiple servers. If the password is compromised, the user must reconfigure the password on all affected servers.
- A password is not transmitted between the client and server using PKI. Instead the sensitive information (private key) is kept on the client. Consequently, the password is less likely to be compromised.

SR OS supports server-sider SSHv2 PKI but does not include a key-generation utility.

Support for PKI should be configured in the system-level configuration, where one or more public keys may be bound to a username. This configuration will not affect any other system security or login functions.

PKI has preference over password or keyboard authentication. PKI is supported using local authentication and using an AAA server with LDAP only. PKI authentication is not supported on TACACS+ or RADIUS, and users with public keys always use local authentication only.

2.5.1.2.1 User public key generation

Before SSH is used with PKI, someone must generate a public/private key pair. This is typically supported by the SSH client software. For example, PuTTY supports a utility called PuTTYGen that will generate key pairs.

SR OS currently supports only RSA and ECDSA user public keys.

If the client is using PuTTY, the user first generates a key pair using PuTTYGen. The user sets the key type to SSH-2 RSA and sets the number of bits to be used for the key. The user can also configure a passphrase, which is used to store the key locally in encrypted form. If configured, the passphrase acts as a password for the private key, and the user must enter it to use the private key. If a passphrase is not configured, the key is stored in plain text locally.

Use the following command to configure the public key for the user on SR OS:

- **MD-CLI**

```
configure system security user-params local-user user public-keys
```

- **classic CLI**

```
configure system security user public-keys
```

On SR OS, the user can program the public key using Telnet/SSH or SNMP.

2.5.1.2.2 Public key authentication

Public key authentication is configured per user. If public key authentication fails, the system offers username and password authentication. If the user does not change the default password and the SSH public key authentication fails, this represents a security risk.

SR OS supports standalone public key authentication without offering username and password authentication using the CLI configuration. This configuration is per user or per system and is disabled by default (that is, the system offers keyboard authentication).

The SR OS server operates as follows:

- By default, it allows public key and password authentication.
- When RADIUS or TACPLUS interactive authentication is enabled, public key and password or keyboard authentication and password is allowed by default.
- The user can enable public key-only authentication per system or per user. This configuration is disabled by default. The user configuration has a higher priority than the system configuration.
- If **interactive-authentication** is enabled in the following contexts the SSH server also accepts interactive keyboard authentication:
  - **MD-CLI**

```
configure system security aaa remote-servers radius
configure system security aaa remote-servers tacplus
```

  - **classic CLI**

```
configure system security radius
```

```
configure system security tacplus
```

- When the **public-key-only** command option is enabled, the server negotiates only the public key method and does not allow password or keyboard interactive login. Even if the password authentication order is “**radius, tacplus, ldap, local**” and the following command is configured, the server does not negotiate or accept keyboard authentication.
  - **MD-CLI**

    ```
    configure system security aaa remote-servers radius interactive-authentication true
    ```

  - **classic CLI**

    ```
    configure system security radius interactive-authentication
    ```

- If **public-key-only** is enabled at the system level, this configuration must be turned off for a specific user to support password or keyword authentication for that user.

2.5.1.2.3 Public key authentication and AAA interaction

When public key-only authentication is enabled, the username and password or keyboard authentication using AAA servers is not allowed, even if the authentication order is set to prefer the AAA server. Only the LDAP server for public key authentication is allowed in this mode.

2.5.1.3 Multichannel SSH

SR OS supports up to 50 open channels per SSH connection.

Use the following command to configure the maximum number of open channels per SSH connection.

```
configure system login-control ssh max-channels-per-connection
```

When the SSH connection has authenticated a user and opened a channel for configuration, another SSH channel is required to retrieve the state information, such as collecting configurations or **show** command output. In this scenario, network managers can set up an additional channel in the existing SSH connection.

Opening a new channel inside an existing authenticated SSH connection mitigates the additional time and memory requirements for establishing a new SSH session. This mitigation is useful when, for example, multiple RPCs from different network manager users to the same device are executed at the same time.

Note: Multiple channels are only supported for SSH and some applications that use SSH as transport. SR OS supports one application with multiple channels over the same connection. Multiple channels are not supported for SFTP or SCP.

2.5.1.4 MAC client and server list

SR OS supports a configurable server and client MAC list for SSHv2. This allows the user to add or remove MAC algorithms from the list. The user can program the strong Hash-based Message Authentication Code (HMAC) algorithms on top of the configurable MAC list (for example, lowest index in

the list) to define the list order presented during algorithm negotiation between the client and server. The first algorithm supported by both the client and the server is selected.

There are two configurable MAC lists:

- server list
- client list

The default MAC list includes all supported algorithms in the following preference order:

1. mac 200 name hmac-sha2-512
2. mac 210 name hmac-sha2-256
3. mac 215 name hmac-sha1
4. mac 220 name hmac-sha1-96
5. mac 225 name hmac-md5
6. mac 240 name hmac-md5-96

2.5.1.5 KEX client and server list

SR OS supports configurable KEX client and server lists. The user can add or remove the needed KEX client or server algorithms to be negotiated using an SSHv2 phase one handshake. The list is an index list, with the lower index having a higher preference in the SSH negotiation. That is, the lowest indexed algorithm in the list is negotiated first in SSH and is at the top of the negotiation list to the peer.

When no KEX list entries are explicitly configured, the system behaves as if the following list is configured:

- kex 170 name ecdh-sha2-nistp521
- kex 180 name ecdh-sha2-nistp384
- kex 190 name ecdh-sha2-nistp256
- kex 200 name diffie-hellman-group16-sha512
- kex 210 name diffie-hellman-group14-sha256
- kex 215 name diffie-hellman-group14-sha1
- kex 220 name diffie-hellman-group-exchange-sha1
- kex 225 name diffie-hellman-group1-sha1

As soon as an algorithm is configured in the KEX list, SR OS starts using the user-defined KEX list instead of the hard-coded list. This means that all default KEX algorithms are removed and only the KEX algorithms configured by the user will be negotiated.

To go back to the hard-coded list, remove all configured KEX indexes until the list is empty. Use the following commands inline with the cipher, MAC server, or client lists:

- **MD-CLI**

```
delete /configure system security ssh client-kex-list-v2
delete /configure system security ssh server-kex-list-v2
```

- **classic CLI**

```
configure system security ssh client-kex-list kex
configure system security ssh client-kex-list no kex
```

```
configure system security ssh server-kex-list kex
configure system security ssh server-kex-list no kex
```

2.5.1.6 Host key algorithm list

SR OS supports the following list of host key algorithms, which are negotiated for the SSH server and SSH client during the SSH handshake:

- ecdsa-sha2-nistp521
- ecdsa-sha2-nistp384
- ecdsa-sha2-nistp256
- rsa-sha2-512
- rsa-sha2-256
- ssh-ed25519
- ssh-rsa

The host key algorithm list is configurable for the server and the client, which allows the user to add or remove host key algorithms from the list. Use the following commands to configure the list:

- **MD-CLI**

```
configure system security ssh server-host-key-list-v2
configure system security ssh client-host-key-list-v2
```

- **classic CLI**

```
configure system security ssh server-host-key-list
configure system security ssh client-host-key-list
```

The algorithm list is an index list, with the lower index having a higher preference in the SSH negotiation. That is, the lowest indexed algorithm in the list is at the top of the negotiation list and is negotiated first in the SSH handshake.

The following table lists the approximate time required to generate the host signature on bootup when the router is acting as the SSH server.

Note: The time required to generate the host signature may vary depending on the power of the CPU component of the router. The estimates in the table should not be used as general guidance for all routers and CPU types.

Table 15: Approximate time to generate the host signature on the SSH server

Algorithm	SIM (ms)	HW (ms)
dsa	130	15000
rsa	60	8000
ecdsa 256	10	1200
ecdsa 384	20	2400
ecdsa 521	10	1200

Algorithm	SIM (ms)	HW (ms)
ed25519	50	6000

Note: Host keys are generated during router bootup and increase the bootup time. To ensure optimized bootup time, the user should disable unwanted host keys.

After upgrading to a software release that supports the new host keys, the user may have to remove the known SSH host key files and database, because the SSH client may negotiate to a more secure and higher preference host key.

2.5.1.7 Regenerate the SSH key without disabling SSH

SR OS supports periodic rollover of the SSH symmetric key. Symmetric key rollover is important in long SSH sessions. Symmetric key rollover ensures that the encryption channel between the client and server is not jeopardized by an external hacker that is trying to break the encryption via a brute force attack.

This feature introduces symmetric key rollover on the SSH client or server. The following are triggers for symmetric key rollover and negotiation:

- the negotiation of the key base on a configured time period
- the negotiation of the key base on a configured data transmission size

For extra security, the key re-exchange is enabled by default under SR OS.

2.5.1.7.1 Key re-exchange procedure

Key re-exchange is initiated by sending an SSH_MSG_KEXINIT packet while a key exchange is not already in progress. When this message is received, a party must respond with its own SSH_MSG_KEXINIT message, except in cases where the received SSH_MSG_KEXINIT was already a reply. Either party may initiate the re-exchange, but the roles must not be changed (for example, the server must remain the server, and the client must remain the client).

Key re-exchange is performed using the encryption that was in effect when the exchange was initiated. Encryption, compression, and MAC methods are not changed before a new SSH_MSG_NEWKEYS is sent after the key exchange (as in the initial key exchange). Re-exchange is processed identically to the initial key exchange, except that the session identifier remains unchanged. Some or all of the algorithms can be changed during the re-exchange. Host keys can also change. All keys and initialization vectors are recomputed after the exchange. Compression and encryption contexts are reset.

RFC 4253 recommends performing a key exchange after every hour or 1Gbytes of transmitted data, which is met by SR OS default implementation.

SR OS can roll over keys via the following mechanisms:

- bytes (default is 1 Gbyte and the keys are negotiated)
- minutes (default is 1 minute)

Note:

- If both the bytes and minutes key rollover mechanisms are configured, the key rollover is based on whichever occurs first.

- If these command options change, only new SSH connections inherit them. The existing SSH connections continue to use the previously configured command options.

2.5.1.8 Cipher client and server list

SR OS supports cipher client and server lists. The user can add or remove the needed SSH cipher client or server algorithms from the negotiation list. The list is indexed with lower indexed algorithms having higher preference in the SSH negotiation. The lowest indexed algorithm in the list is negotiated first in SSH and is on top of the negotiation list to the peer.

The default server and client lists for SSHv2 include all supported algorithms with the following preference:

- cipher 190 name aes256-ctr
- cipher 192 name aes192-ctr
- cipher 194 name aes128-ctr
- cipher 200 name aes128-cbc
- cipher 205 name 3des-cbc
- cipher 225 name aes192-cbc
- cipher 230 name aes256-cbc

2.5.1.9 Static host public key list

SR OS supports the configuration of a static host public key list to use when acting as an SSH client. The static host key list allows the SR OS client to SSH to an SFTP server without needing to accept a CLI fingerprint prompt as the host key can be entered manually via CLI. This allows SR OS to operate with SSH-based applications that do not use CLI, such as SFTP for secure transmission, and therefore cannot display the fingerprint prompt.

Use the commands in the following context to configure the list of SSH host public keys.

```
configure system security ssh client-known-hosts
```

The user can configure multiple static host public keys for a host.

Static keys are stored in the same list as dynamically learned keys, with static key entries added to the bottom of the list. SR OS supports up to 32 dynamic keys and 32 static keys in the list. A host can have both static and dynamic keys associated with it simultaneously. When new static keys are added to the list, they do not overwrite dynamic keys, even if the entries for both keys match.

The SR OS behavior when connecting to a server is as follows:

- If SR OS finds a matching key entry for the host in the host public key list, it accepts the connection without requiring a fingerprint prompt.
- If SR OS does not find a matching key entry in the list, one of the following occurs:
  - For applications without CLI access, SR OS drops the connection to the host.
  - For applications with CLI access, SR OS displays a fingerprint prompt in the CLI. If the user accepts the prompt, SR OS adds a new dynamic key to the public host key list and connects to the host.

Use the following command to display the list of known dynamic and static host public keys.

```
show system security ssh client-known-hosts
```

2.5.1.10 SSH session closing behavior

The SSH session closes automatically when the channel opened last in the session is closed.

The SSH keepalive intervals are disabled on SR OS, which means the following:

- The SR OS SSH server does not close the session when the client SSH keepalive intervals time out.
- The client SSH keepalive intervals cannot be used to keep the connection to the SR OS server open.

2.5.1.11 CLI and SNMP considerations

Support for SSH version 1 (SSHv1) has been removed from SR OS. The following considerations apply when upgrading from a pre-22.10 release.

Note: This information applies for the classic CLI.

An In-Service Software Upgrade (ISSU) conversion must be performed to filter previously existing ciphers and set the SSH version to **2**. The customers using version 1 are switched to version 2 with default ciphers and HMAC algorithms. The default cipher set is auto-created.

SNMP

All SNMP objects remain intact. Although SSH version 1 is not visible in the **info** or **show** commands, the SNMP objects are present in the MIB and cannot be set. Attempting to configure blocked version 2 ciphers or version 1 settings returns an error.

2.5.2 Exponential login backoff

A malicious user may attempt to gain CLI access by means of a dictionary attack, whereby a script is used to automatically attempt to log in as an `admin` user, and a dictionary list is used to test all possible passwords. Using the exponential-backoff feature in the **configure system login-control** context, the SR OS increases the delay between login attempts exponentially to mitigate attacks.

When a user tries to log in to a router using a Telnet or an SSH session, there are a limited number of attempts allowed to authenticate a user. The interval between the unsuccessful attempts change after each attempt, at 1, 2, and 4-second intervals. If the system is configured for user lockout, the user will be locked out when the number of attempts is exceeded.

However, if lockout is not configured, there are three password entry attempts allowed after the first failure, at 1, 2, and 4-second intervals, in the first session, and then the session terminates. Users do not have an unlimited number of login attempts per session. After each failed authentication, the wait period becomes longer until the maximum number of attempts is reached.

The SR OS terminates after four unsuccessful attempts. A wait period is never longer than 4 seconds. The periods are fixed and restart in subsequent sessions.

Use the following system-wide configuration commands together to mitigate attacks:

- **MD-CLI**

```
configure system login-control exponential-backoff
configure system security user-params attempts
```

- **classic CLI**

```
configure system login-control exponential-backoff
configure system security password attempts
```

Exponential backoff applies to any user and by any login method such as console, SSH, and Telnet.

See Configuring login controls for more information.

2.5.3 User lockout

When a user exceeds the maximum number of attempts allowed (the default is 3 attempts) during a specific period of time (the default is 5 minutes), the account used during those attempts are locked out for a pre-configured lock-out period (the default is 10 minutes).

A security or LI event log is generated as soon as a user account has exceeded the number of allowed attempts. Use the following command to display the total number of failed attempts per user.

```
show system security user
```

In addition to the security or LI event log, an SNMP trap is also generated so that any SNMP server (including the NSP NFM-P) can use the trap for an action. The account is automatically re-enabled as soon as the lock-out period has expired.

Use the following command to display a list of users who are currently locked out.

```
show system security user lockout
```

Use the following command with the applicable option to clear a lockout for a specific **user** or **all** users:

- **MD-CLI**

```
admin clear security lockout
```

- **classic CLI**

```
admin clear lockout
```

2.5.4 CLI login scripts

The SR OS supports automatic execution of CLI scripts when a user successfully logs into the router and starts a CLI session.

Use the following command to configure a login script for users who authenticate using the local user database:

- **MD-CLI**

```
configure system security user-params local-user user console login-exec file-url
```

- **classic CLI**

```
configure system security user console login-exec file-url
```

You can configure a global login-script to execute a common script when any user logs into the CLI. You can also configure a per-user login-script to execute when a specific user logs into the CLI. These login-scripts execute when the user is authenticated using the local user database, TACACS+, or RADIUS.

Use the following command to configure a global login script:

- **MD-CLI**

```
configure system login-control login-scripts global-script global-login-script-url
```

- **classic CLI**

```
configure system login-control login-scripts global global-login-script-url
```

Use the following command to configure a user-specific login script pointing to a user-defined directory:

- **MD-CLI**

```
configure system login-control login-scripts per-user-script user-directory dir-url
```

- **classic CLI**

```
configure system login-control login-scripts per-user user-directory dir-url file-name file-name
```

2.5.5 File access controls

Access files on the router locally using the CLI **file** commands and output modifiers, such as **>** (file redirect), or remotely via FTP and SCP. SR OS provides file access controls that allow an administrator to create users with different permissions; for example:

- access to all files and the ability to save the configuration
- access to home directory files only and the ability to save the configuration
- access to home directory files only and no ability to save the configuration
- no file access and no ability to save the configuration

The file access controls provide different levels of user access depending on user privileges. File access controls can also be configured to allow users to save the configuration to a system file that is stored outside their home directory when their file access is restricted to their home directory.

A home directory is typically a working space for the user, for example `cf3:\users\user1`. Although the home directory can be configured to contain saved configuration files, log files, or other system files.

Note: Take care to only configure home directory for users who are intended to have access to those files.

The following controls affect file access for local or remote users, and can be set via the CLI, TACACS+ VSAs, or RADIUS VSAs:

- restricted to home – restrict file access to the local home directory of the user
- home directory – home directory for the user. Nokia recommends users do not configure this command option in the TACACS+ or RADIUS default template because each user should have their own home directory.
- save when restricted – save configurations when the user is restricted to their home directory, even if the saved configuration file is outside the home directory of the user

High-privilege users and administrators have access to the router configuration file via the CLI, SCP, and SFTP, and must be trusted.

Medium-privilege and low-privilege users with access to a subset of the configuration must be configured with “restricted to home” and a “home directory” to restrict their access to the file system via the CLI, SCP, and SFTP. The “home directory” must not contain the saved configuration file.

The following table lists examples of various types of users and privileges with the corresponding file access control configuration.

Table 16: File access control configuration

Type of user	File system access	Configuration access	Restricted to home	Home directory	Save when restricted
High-privilege administrator	Full	Yes	Disabled (false)	Unconfigured	Unconfigured
High-privilege user with access to all configuration	Private directory with environment settings and Python applications	Yes	Enabled (true)	cf3:\users\user1	Enabled (true)
Medium-privilege user with access to a subset of configuration using AAA command authorization	Directory of CLI scripts and Python applications	Yes	Enabled (true)	cf3:\scripts\ script-group-a	Enabled (true)
Medium-privilege user with access to a subset of configuration using AAA command authorization	None	Yes	Enabled (true)	Unconfigured	Enabled (true)

Type of user	File system access	Configuration access	Restricted to home	Home directory	Save when restricted
Low-privilege operational user	Private directory with environment settings and operational scripts	No	Enabled (true)	cf3:\users\user2	Unconfigured[2]
Low-privilege operational user	None	No	Enabled (true)	Unconfigured	Unconfigured[2]
Low-privilege user for managing XML accounting files	Accounting directory	No	Enabled (true)	cf2:\act	Unconfigured[2]
Low-privilege user for managing log files	Log directory	No	Enabled (true)	cf2:\log	Unconfigured[2]

The following examples show how to use the CLI to configure different permissions for local users. The administrator must create a home directory for each user.

Example: Access to all files and the ability to save the configuration

Use the following configuration for a high-privilege administrator that needs access to all files:

- **MD-CLI**

```
configure system security user-params local-user user user1 restricted-to-home false
```

- **classic CLI**

```
configure system security user user1 no restricted-to-home
```

Example: Access to home directory files only and the ability to save the configuration

Use the following configuration for a medium-privilege user who can access some parts of the configuration and needs to save the configuration, but is denied access to other parts of the configuration in a AAA profile:

- **MD-CLI**

```
configure system security user-params local-user user user2 home-directory "cf3:\users\
user2"
configure system security user-params local-user user user2 restricted-to-home true
configure system security user-params local-user user user2 save-when-restricted true
```

[2] Configuration save operations (for example, **admin save**) are controlled by AAA command authorization. Optionally, “save when restricted” can be disabled to explicitly deny configuration save operations for these user types.

- **classic CLI**

```
configure system security user user2 home-directory "cf3:\users\user2"
configure system security user user2 restricted-to-home
configure system security user user2 save-when-restricted
```

Use a similar configuration for a user account used to collect XML accounting files. In this case, the **home-directory** is set to the location where completed accounting files are written, for example, `cf3:/act`.

Example: Access to home directory files only and no ability to save the configuration

Use the following configuration for a low-privilege user who does not have access to any part of the configuration, but still requires a working area on the local file system for their own files:

- **MD-CLI**

```
configure system security user-params local-user user user3 home-directory "cf3:\users\
user3"
configure system security user-params local-user user user3 restricted-to-home true
configure system security user-params local-user user user3 save-when-restricted false
```

- **classic CLI**

```
configure system security user user3 home-directory "cf3:\users\user3"
configure system security user user3 restricted-to-home
configure system security user user3 no save-when-restricted
```

Example: No file access and no ability to save the configuration

Use the following configuration for a low-privilege user who does not have access to any part of the configuration, and does not require any file system access:

- **MD-CLI**

```
configure system security user-params local-user user user4 restricted-to-home true
configure system security user-params local-user user user4 save-when-restricted false
```

- **classic CLI**

```
configure system security user user4 restricted-to-home
configure system security user user4 no save-when-restricted
```

For configuration examples using RADIUS VSAs, see RADIUS configuration for file access control using VSAs and for configuration examples using TACACS+ VSAs, see TACACS+ configuration for file access control using VSAs.

2.5.6 802.1x network access control

The SR OS supports network access control of client devices (such as PCs or STBs) on an Ethernet network using the IEEE 802.1x standard. IEEE 802.1x is known as Extensible Authentication Protocol (EAP) over a LAN network or (EAPOL).

2.5.7 TCP Enhanced Authentication Option

The TCP authentication mechanism authenticates the source of TCP packets at the TCP Layer, blocking packets from unauthorized sources. TCP authentication improves security for TCP-based protocols, such as BGP and LDP, and is used for applications where secure administrative access to both the endpoints of the TCP connection is normally available.

TCP authentication provides enhanced security and key management, protecting against collision attacks and replay attacks. This mechanism uses strong MACs and a larger range of authentication algorithms to improve security. TCP authentication allows for keys to change within an active TCP session using the receive next key ID (RNextKeyID) field without tearing down the TCP session.

Two types of TCP authentication exist with different interoperability:

- TCP Authentication Option (TCP-AO)
- TCP Enhanced Authentication

TCP-AO

TCP-AO is defined in RFC 5925, *The TCP Authentication Option*, and RFC 5926, *Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)*. TCP-AO replaces TCP MD5, which is defined in RFC 2385, *Protection of BGP Sessions via the TCP MD5 Signature Option*. TCP-AO is the interoperable standard.

TCP Enhanced Authentication

TCP Enhanced Authentication is defined in *draft-bonica-tcp-auth-04*. It is only supported on SR OS devices and it is not interoperable.

Note: TCP Enhanced Authentication predates RFC 5925 and is not recommended for new installations.

2.5.7.1 Packet formats

TCP-AO packet format

TCP-AO packet format details are described in the RFC 5925.

The following commands must have the **tcp-ao** option configured to enable TCP-AO:

- **MD-CLI**

```
configure system security keychains keychain tcp-option-number receive
configure system security keychains keychain tcp-option-number send
```

- **classic CLI**

```
configure system security keychain tcp-option-number receive
configure system security keychain tcp-option-number send
```

The following figure shows the packet format for TCP-AO.

Figure 7: TCP-AO packet format

The following list describes the preceding TCP-AO fields:

- **Kind (1 byte)**

  The Kind field indicates TCP-AO and uses a value of 29.

- **Length (1 byte)**

  The Length field indicates the length of the option in bytes and includes the Kind, Length, KeyID, RNextKeyID, and MAC fields.

  The value must be equal to or greater than 4.

- **KeyID (1 byte)**

  The KeyID field identifies the Master Key Tuple (MKT) used to generate traffic keys, which are then used to generate the MAC for segment authentication. The KeyID comes from the **entry** number in the configuration.

- **RNextKeyID (1 byte)**

  The RNextKeyID field indicates the receive next key ID, which is the MKT ready at the sender to authenticate segments.

- **MAC**

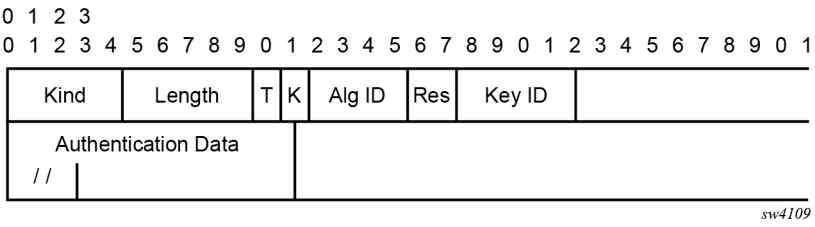
  The MAC field is the message authentication code. All the TCP options are included in the MAC calculation.

TCP Enhanced Authentication packet format

TCP Enhanced Authentication packet format details are described in *Authentication for TCP-based Routing and Management Protocols draft-bonica-tcp-auth-04.txt*.

The following figure shows the TCP Enhanced Authentication packet format.

Figure 8: Packet format



The following list describes the preceding fields:

- **Kind (8 bits)**

  The Kind field specifies the TCP Enhanced Authentication option. The value used here comes from the keychain configuration **tcp-option-number**. The **send** value, which can be configured as **253** or **254**,

populates the Kind field on outgoing packets The **receive** value, which can be configured as **253**, **254** or **253&254**, is checked on the received packets to identify the TCP Enhanced Authentication option.

- **Length (8 bits)**

  The Length field specifies the length of the TCP Enhanced Authentication option. The Length field for SR OS is always set to a value of 16.

- **T-bit (1 bit)**

  The T-bit indicates the TCP options in the TCP header as part of the MAC calculation. The T-bit is always set to 0 for SR OS, which means all the TCP options are included in the MAC calculation.

- **K-bit (1 bit)**

  The K-bit field is reserved for future enhancement and the value is zero.

- **Alg ID (6 bits)**

  The Alg ID field specifies the MAC algorithm.

- **Res (2 bits)**

  The Res field has reserved bits and this must be set to zero.

- **KeyID (6 bits)**

  The KeyID field comes from the **entry** number in the keychain configuration.

- **Authentication Data (variable length)**

  The Authentication Data field contains the MAC or Message Authentication Code.

2.5.7.2 Keychain

TCP authentication, which includes TCP-AO and TCP Enhanced Authentication, uses keychains that are associated with every protected TCP connection. The keychain mechanism is supported by the following protocols:

- BGP
- LDP
- OSPF
- IS-IS
- RSVP-TE
- NTP
- MCS
- PCEP

The keychain mechanism enables the creation of keys used to authenticate protocol communications. Keychains allow users to automatically change the key used, the algorithm used, or both at specific times. Users can schedule new keys in advance and the system will start using the keys, algorithm, or both at the specified time. Systems using keychains must be synchronized in time, using, for example, NTP.

Each keychain entry defines the authentication attributes used in authenticating protocol messages from remote peers or neighbors. The configuration must include at least one key entry to be valid. Through the use of the keychain mechanism, authentication keys can be changed without affecting the state of the associated protocol adjacencies.

Each key within a keychain must include the following attributes for the authentication of protocol messages:

- key identifier – unique identifier, expressed as a decimal integer
- authentication algorithm – see Table 18: Security algorithm support per protocol
- authentication key – key used by the authentication algorithm to authenticate packets
- direction – packet stream direction in which the key is applied (receive direction, send direction, or both)
- begin time – time at which a new authentication key can be used

Optionally, each key can include the following attributes:

- end time – time at which the authentication key becomes inactive (applies to received packets only)
- tolerance – period in which both old and new authentication key values can overlap and both keys are allowed on received packets (applies to received packets only)

BGP and LDP do not support MD5 via keychains. Both of these protocols support MD5 authentication using the *authentication-key* parameter. Nokia recommends users only configure either the **authentication-key** or the **auth-keychain** parameter.

The algorithms supported by all TCP-based protocols that use keychains (BGP, LDP, PCEP) are:

- AES-CMAC-128-96
- HMAC-SHA-1-96
- HMAC-SHA-256-96
- HMAC-SHA-256-128

Although a key in a keychain could be configured with some of the other algorithms, the TCP Layer treats that key as expired and does not use it.

Note: Although the algorithms HMAC-SHA-256, HMAC-SHA-256-128, and HMAC-SHA-256-96 use the same basic algorithm to calculate the hash, the output produced is not the same. The three names are not equivalent or interchangeable. The same applies to HMAC-SHA-1-96 and HMAC-SHA-1.

The **entry** number populates the KeyID fields and these numbers must match on both ends for a successful TCP connection.

The following table shows the mapping between these attributes and the CLI command to set them.

The following table shows the mapping between these attributes and the CLI command to set them.

Table 17: Keychain mapping

Definition	Configuration

The key identifier expressed as an integer (0...63)	**MD-CLI**
	`configure system security keychains keychain bidirectional entry`
	`configure system security keychains keychain receive entry`
	`configure system security keychains keychain send entry`
	classic CLI
	`configure system security keychain direction bi entry`
	`configure system security keychain direction uni receive entry`

Definition	Configuration
	`configure system security keychain direction uni send entry`

Authentication algorithm to use with key[i]	**MD-CLI**
	`configure system security keychains keychain bidirectional entry algorithm`
	`configure system security keychains keychain receive entry algorithm`
	`configure system security keychains keychain send entry algorithm`
	classic CLI
	`configure system security keychain direction bi entry key algorithm`
	`configure system security keychain direction uni receive entry key algorithm`
	`configure system security keychain direction uni send entry key algorithm`

Shared secret to use with key[i]	Use the shared secret with the following:
	• **MD-CLI**
	`configure system security keychains keychain bidirectional entry`
	`configure system security keychains keychain receive entry`
	`configure system security keychains keychain send entry`
	• **classic CLI**
	`configure system security keychain direction bi entry`
	`configure system security keychain direction uni receive entry`
	`configure system security keychain direction uni send entry`

A vector that determines whether the key[i] is to be used to generate MACs for inbound segments, outbound segments, or both	**MD-CLI**
	`configure system security keychains keychain bidirectional entry`
	`configure system security keychains keychain receive entry`
	`configure system security keychains keychain send entry`
	classic CLI
	`configure system security keychain direction bi entry`
	`configure system security keychain direction uni receive entry`
	`configure system security keychain direction uni send entry`

Start time from which key[i] can be used	**MD-CLI**
	`configure system security keychains keychain bidirectional entry begin-time`
	`configure system security keychains keychain send entry begin-time`
	classic CLI
	`configure system security keychain direction bi entry begin-time`
	`configure system security keychain direction uni send entry begin-time`

Definition	Configuration
End time after which key[i] cannot be used by sending TCPs	Inferred by the begin-time of the next key (youngest key rule)

Start time from which key[i] can be used	**MD-CLI**
	`configure system security keychains keychain bidirectional entry begin-time`
	`configure system security keychains keychain bidirectional entry tolerance`
	`configure system security keychains keychain receive entry begin-time`
	`configure system security keychains keychain receive entry tolerance`
	classic CLI
	`configure system security keychain direction bi entry begin-time`
	`configure system security keychain direction bi entry tolerance`
	`configure system security keychain direction uni receive entry begin-time`
	`configure system security keychain direction uni receive entry tolerance`

End time after which key[i] cannot be used	**MD-CLI**
	`configure system security keychains keychain receive entry end-time`
	classic CLI
	`configure system security keychain direction uni receive entry end-time`

The following table lists the authentication algorithms that can be used in association with specific routing protocols.

Note: Other vendors may use other names for the same algorithm. Always verify the correct algorithm names in the vendor's documentation.

Table 18: Security algorithm support per protocol

Protocol	password (cleartext)	message-digest (MD5)	hmac-md5	hmac-sha-1-96	hmac-sha-1	hmac-sha-256	hmac-sha-256-96	hmac-sha-256-128	aes-128-gcm-16	aes-128-cmac-96	aes-128-cmac-128
OSPF	Yes	Yes	—	Yes	Yes	Yes	—	—	—	—	—
IS-IS	Yes	—	Yes	—	Yes	Yes	—	—	—	—	—
RSVP	—	—	Yes	Yes	Yes	Yes	—	—	—	—	—

Protocol	password (cleartext)	message-digest (MD5)	hmac-md5	hmac-sha-1-96	hmac-sha-1	hmac-sha-256	hmac-sha-256-96	hmac-sha-256-128	aes-128-gcm-16	aes-128-cmac-96	aes-128-cmac-128
BGP	—	Yes [3]	—	Yes	—	—	Yes	Yes	—	Yes	—
LDP	—	Yes [3]	—	Yes	—	—	Yes	Yes	—	Yes	—
NTP	—	—	—	—	—	—	—	—	—	—	Yes
PCEP	—	—	—	Yes	—	—	Yes	Yes	—	Yes	—
Multi-chassis redundancy	—	—	—	—	—	—	—	—	Yes	—	—

The following table lists the digest lengths generated by each algorithm.

Table 19: Digest length for each algorithm

	password (cleartext)	message-digest (MD5)	hmac-md5	hmac-sha-1-96	hmac-sha-1	hmac-sha-256	hmac-sha-256-96	hmac-sha-256-128	aes-128-gcm-16	aes-128-cmac-96	aes-128-cmac-128
Digest length (bytes)	N/A	16	16	12	20	32	12	16	16	12	16

2.5.7.3 Keychain configuration guidelines and behaviors

Consider the following keychain configuration guidelines and behaviors:

- Use either the **authentication-key** command or the **authentication-keychain** command in the MD-CLI (**auth-keychain** command in the classic CLI) with the protocols. Both commands cannot be supported at the same time. If both commands are configured, the system applies the **authentication-keychain** configuration in the MD-CLI (**auth-keychain** configuration in the classic CLI) and ignores the **authentication-key** command.
- A keychain cannot be referenced by a protocol until it has been configured.
- If a keychain is referenced by a protocol, the keychain cannot be deleted.

[3] MD5 is not supported by BGP and LDP on keychains. Use the *authentication-key* parameters under **bgp** and **ldp** to use MD5 authentication.

- If multiple keys in a keychain are valid at the same time, the newest key (key with the most current **begin-time**) is used.
- If a protocol sends a packet that is configured to use a keychain, the most current key from that keychain is used.
- If a protocol receives a packet that is configured to use a keychain, the current key set is returned to authenticate the received packet.
  - The key set includes the currently active keys (based on the current system time) and the begin or end time associated with each key in the specified keychain.
  - If a tolerance value is set for a key, the key is returned as part of the key set if the current time is within the begin time of the key, plus or minus the tolerance value. For example, if the **begin-time** is 12:00 p.m. and the **tolerance** is 600 seconds, the new key is included from 11:55 a.m. and the key to be replaced is included until 12:05 p.m.
- The **end-time** and **tolerance** command options apply only to received packets. Transmitted packets always use the newest key, regardless of the tolerance value.
- If a keychain exists but there are no active key entries with an authentication type that matches the type supported by the protocol, inbound protocol packets are not authenticated and are discarded, and no outbound protocol packets are sent.
- If a keychain exists but the last key entry has expired, a log entry is raised indicating that all keychain entries have expired.
- Special cases:
  - The OSPF and RSVP-TE protocols continue to authenticate inbound and outbound traffic using the last valid authentication key.
  - The IS-IS protocol does not revert to an unauthenticated state and requires that the old key is not used; therefore, when the last key has expired, all traffic is discarded.

For information about associating keychains with protocols, see:

- *7705 SAR Gen 2 Router Configuration Guide*
- *7705 SAR Gen 2 MPLS Guide*
- *7705 SAR Gen 2 Services Overview Guide*

2.5.7.4 Key rollover

Use the following commands to configure keychain authentication to rollover to a new key without tearing down the session:

- **MD-CLI**

```
configure system security keychains keychain receive entry begin-time
configure system security keychains keychain receive entry end-time
configure system security keychains keychain send entry begin-time
configure system security keychains keychain bidirectional entry begin-time
```

- **classic CLI**

```
configure system security keychain direction uni receive entry begin-time
configure system security keychain direction uni receive entry end-time
configure system security keychain direction uni send entry begin-time
```

```
configure system security keychain direction bi entry begin-time
```

The **begin-time** command configures when the authentication key starts to authenticate the protocol stream and the **end-time** command configures when the authentication key is no longer eligible to authenticate the protocol stream. The system uses the key configured with the most recent **begin-time**. For more information about configuring keychain authentication, see Configuring keychain authentication. A user does not have to configure an **end-time**, and can instead choose to configure multiple entries with different **begin-time** configurations so that the newest key (key with the most current **begin-time**) is used. The current key pair is replaced to improve security without a session loss between two peers.

2.5.8 gRPC authentication

gRPC communication between the client and server must be authenticated and encrypted. The following types of authentication are supported:

- **Authentication via session credentials**

  Session credentials operate similarly to device authentication, ensuring that the device is allowed in the network and authorized by the provider. This type of authentication is performed using PKI and X.509.3 certificates. gRPC uses TLS for session authentication. The SR OS supports TLS servers for gRPC.

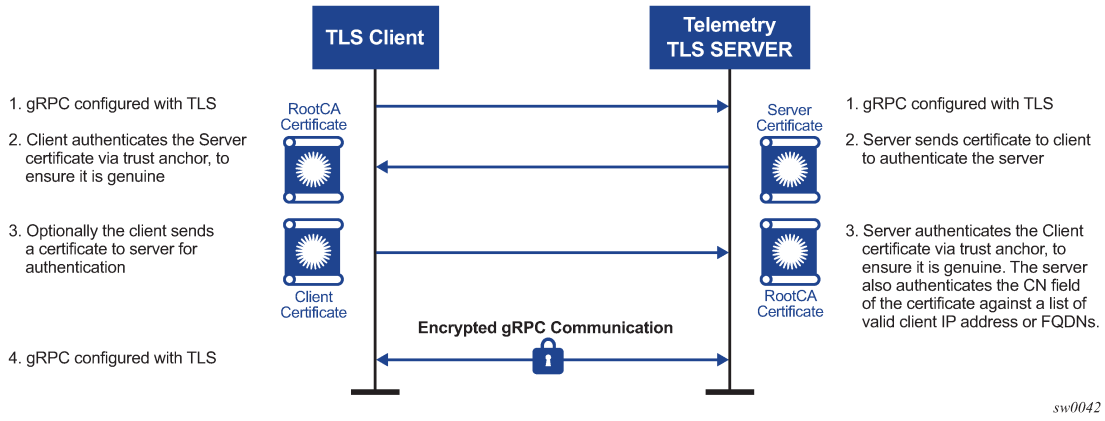
- **Authentication using channel credentials**

  Channel credentials use a username and password that are entered at the gRPC client terminal to authenticate gRPC packets using an AAA method.

Session authentication provides proof that the client and server are authorized devices and that they belong to the provider. After authentication, the session becomes encrypted using TLS, and gRPC PDUs are transmitted between the client and server.

The following figure shows a basic session authentication using TLS.

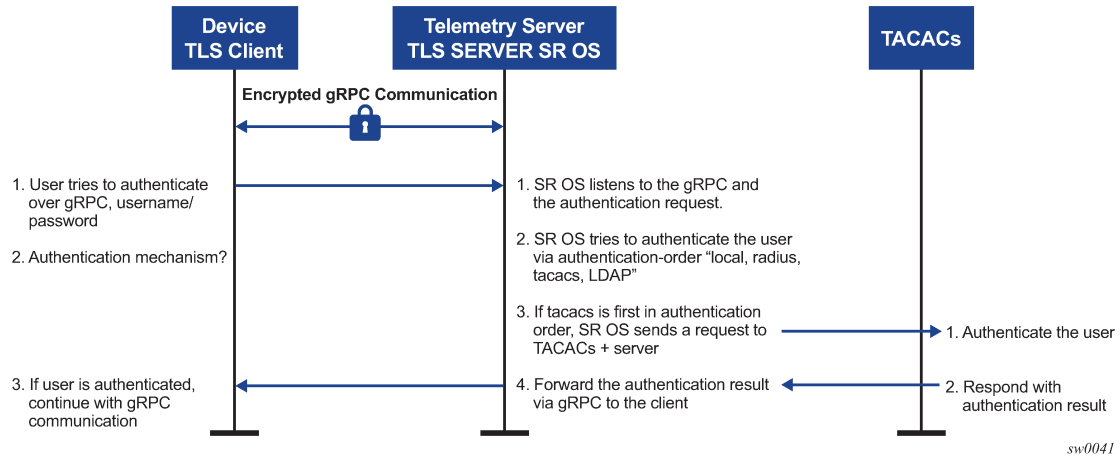
Figure 9: Session authentication using TLS



Channel credentials use username and password authentication. Each gRPC channel packet can contain a username and a password. Authentication is done through standard SR OS authentication order and mechanisms. All current authentication methods, including local and AAA servers, are applicable to gRPC channels. In addition, all authentication orders currently used by Telnet or SSH are compatible with gNMI call authentication.

The following figure shows a basic gNMI call authentication using the SR OS.

Figure 10: gNMI call authentication using SR OS



gRPC channel packets contain the username and password in clear text and are only encrypted using TLS. If a TLS server profile is assigned to the gRPC session, all PDUs between the server and client are encrypted. If TLS becomes operationally down, no gRPC PDUs are transmitted in clear text.

The SR OS relies on existing authentication mechanisms for gRPC channels, including:

- AAA servers and local authentication orders configured using the following commands:
  - **MD-CLI**

    ```
    configure system security user-params authentication-order
    ```

  - **classic CLI**

    ```
    configure system security password authentication-order
    ```

- password complexity rules
- requiring the user to be configured as part of gRPC access by using the following commands:
  - **MD-CLI**

    ```
    configure system security user-params local-user user access grpc
    ```

  - **classic CLI**

    ```
    configure system security user access grpc
    ```

- disconnecting the gRPC session by using the **admin disconnect** command

Note: The gRPC is not affected by password aging.

Security profiles can authorize bulk **get**, **set**, and **subscribe** gRPC commands that are received by the server. Profiles can be configured to permit or deny specific gRPC commands; for example, a profile for one user can authorize **get** and **set** commands, while a profile for another user can authorize only **get** commands.

2.5.9 Hash management per management interface configuration

Hash management is configurable per management interface for example, for the following:

- MD-CLI
- classic CLI
- NETCONF
- gRPC

Each management interface has its own write-hash algorithm. Depending on which management interface the user logs into, the write hash of that interface should be checked and used for displaying the critical phrases.

In the classic CLI interface, the read and write hash algorithms can be different, for example, hash for write and hash2 for read.

In the MD-CLI, NETCONF, and gRPC interfaces, when a hash is configured, only write is implemented using that hash algorithm. For example, if hash2 is configured, SR OS displays the phrase in hash2 format and reads the phrase in all formats. The read algorithm is not affected by hash algorithm configuration and SR OS reads in all hash formats.

2.5.9.1 Hash encryption using AES-256

Hash and hash2 use the AES-256 algorithm for all interfaces. However, hash2 uses module-specific text to make the hash unique per module or protocol. For example, BGP uses a different pre-pending text than IGP. This pre-pending text is appended to the key and then hashed using hash2.

In the classic CLI, hash has been changed to AES-256. Upgrade from DES to AES-256 is allowed and loading a config file in classic CLI with DES to a new software that supports AES-256 is also allowed.

The DES and the DES key should only be used for decryption of the old password to obtain clear text and the password should then be rehashed using AES-256. The few characters of the old hashed phrase are used to determine that the phrase is hashed using DES.

2.5.9.2 Cleartext

The cleartext option for the write algorithm displays the hash in clear text in the config file, info, info detail, and so on.

2.6 Configuring security with CLI

This section provides information to configure security using the command line interface.

2.6.1 Configuring management access filters

The following example shows a management access filter configuration that accepts packets matching the criteria specified in IP, IPv6, and MAC entries. Non-matching packets are denied.

Example: MD-CLI

```
[ex:/configure system security management-access-filter]
A:admin@node-2# info
    ip-filter {
        default-action reject
        entry 10 {
            description "Accept SSH from mgmnt subnet"
            action accept
            match {
                protocol tcp
                src-ip {
                    address 192.168.5.0/26
                }
                dst-port {
                    port 22
                    mask 65535
                }
            }
        }
    }
    ipv6-filter {
        default-action accept
        entry 10 {
            action reject
            log-events true
            match {
                next-header rsvp
                src-ip {
                    address 2001:db8:1000::/64
                }
            }
        }
    }
    mac-filter {
        default-action accept
        entry 12 {
            action accept
            match {
                service "1"
                frame-type ethernet-ii
                src-mac {
                    address 00:01:01:01:01:01
                    mask ff:ff:ff:ff:ff:ff
                }
            }
        }
    }
```

Example: classic CLI

```
A:node-2>config>system>security>mgmt-access-filter# info
----------------------------------------------
                ip-filter
                    default-action deny
                    entry 10
                        description "Accept SSH from mgmnt subnet"
                        src-ip 192.168.5.0/26
                        protocol tcp
                        dst-port 22 65535
                        action permit
                    exit
                exit
```

```
                ipv6-filter
                    default-action permit
                    entry 10
                        src-ip 2001:db8:1000::/64
                        next-header rsvp
                        log
                        action deny
                    exit
                exit
                mac-filter
                    default-action permit
                    entry 12
                        match frame-type ethernet_II
                            svc-id 1
                            src-mac 00:01:01:01:01:01 ff:ff:ff:ff:ff:ff
                        exit
                        action permit
                    exit
                exit
-----------------------------------------------
```

2.6.2 Configuring the IPsec certificate

The following example shows the importation of a certificate from a PEM format.

Example: Importing a certificate from a PEM format (MD-CLI)

```
[/admin system security pki]
A:admin@Dut-AF# import type certificate input-url cf3:/pre-import/R1-0cert.pem format pem
 output-file R1-0cert-der
```

Example: Importing a certificate from a PEM format (classic CLI)

```
A:node-2# admin certificate import type cert input cf3:/pre-import/R1-0cert.pem output R1-
0cert.der format pem
```

The following example shows the exportation of a certificate to PEM format.

Example: Exporting a certificate to a PEM format (MD-CLI)

```
[/admin system security pki]
A:admin@node-2# export type certificate input-file R1-0cert.der format pem output-url
 cf3:/R1-0cert.pem
```

Example: Exporting a certificate to a PEM format (classic CLI)

```
A:node-2# admin certificate export type cert input R1-0cert.der output cf3:/R1-0cert.pem
 format pem
```

The following example shows certificate authority (CA) profile configuration.

Example: CA profile configuration (MD-CLI)

```
[ex:/configure system security pki]
A:admin@node-2# info
    ca-profile "Root" {
        admin-state enable
```

```
        description "Root CA"
        cert-file "R1-0cert.der"
        crl-file "R1-0crl.der"
    }
```

Example: CA profile configuration (classic CLI)

```
A:node-2>config>system>security>pki# info
----------------------------------------------
                ca-profile "Root" create
                    description "Root CA"
                    cert-file "R1-0cert.der"
                    crl-file "R1-0crl.der"
                    no shutdown
                exit
----------------------------------------------
```

The following example shows IKE policy with certificate authority configuration.

Example: IKE policy with certificate authority configuration (MD-CLI)

```
[ex:/configure ipsec ike-policy 1]
A:admin@node-2# info
    ike-version-2 {
        auth-method cert
        own-auth-method psk
    }
```

Example: IKE policy with certificate authority configuration (classic CLI)

```
A:node-2>config>ipsec>ike-policy# info
----------------------------------------------
            ike-version 2
            auth-method cert-auth
            own-auth-method psk
----------------------------------------------
```

The following example shows a static LAN-to-LAN using certificate authority.

Example: Static LAN-to-LAN using certificate authority configuration (MD-CLI)

```
[ex:/configure service vprn "new"]
A:admin@node-2# info
    interface "VPRN1" {
        tunnel true
        sap tunnel-1.private:1 {
            ipsec-tunnel "Sanity-1" {
                admin-state enable
                key-exchange {
                    dynamic {
                        ike-policy 1
                        ipsec-transform [1]
                        pre-shared-key "qGAaHFZ9e/YXSsSCzlYbYWetW+Md2bDwwA== hash2"
                        cert {
                            cert-profile "M2cert.der"
                            trust-anchor-profile "R1-0"
                        }
                    }
                }
                tunnel-endpoint {
                    local-gateway-address 10.1.1.13
```

```
                    remote-ip-address 10.1.1.15
                    delivery-service "300"
                }
                security-policy {
                    id 1
                }
            }
        }
    }
```

Example: Static LAN-to-LAN using certificate authority configuration (classic CLI)

```
A:node-2>config>service>vprn# info
----------------------------------------------
    interface "VPRN1" tunnel create
        sap tunnel-1.private:1 create
            ipsec-tunnel "Sanity-1" create
                security-policy 1
                local-gateway-address 10.1.1.13 peer 10.1.1.15 delivery-service 300
                dynamic-keying
                    ike-policy 1
                    pre-shared-key "Sanity-1"
                    transform 1
                    cert
                        trust-anchor "R1-0"
                        cert "M2cert.der"
                        key "M2key.der"
                    exit
                exit
                no shutdown
            exit
        exit
    exit
----------------------------------------------
```

2.6.3 Configuring local command authorization profiles

Profiles are used to deny or permit access to a hierarchical branch or specific commands.

The following example output displays a local command authorization profile called "service-22" that is associated with a username "user1".

Example: MD-CLI

```
[ex:/configure system security aaa local-profiles]
A:admin@node-2# info
    profile "service-22" {
        default-action permit-all
        entry 1 {
            match "configure"
            action permit
        }
        entry 2 {
            match "configure service vprn <22>"
            action read-only
        }
        entry 3 {
            match "show"
        }
        entry 4 {
```

```
            match "exit"
        }
    }
```

```
[ex:/configure system security user-params local-user]
A:admin@node-2# info
    user "user1" {
        console {
            member ["service-22"]
        }
    }
```

Example: classic CLI

```
A:node-2>config>system>security# info
----------------------------------------------
...
            profile "service-22"
                default-action permit-all
                entry 1
                    match "configure"
                    action permit
                exit
                entry 2
                    match "configure service vprn <22>"
                    action read-only
                exit
                entry 3
                    match "show"
                exit
                entry 4
                    match "exit"
                exit
            exit
...
```

```
----------------------------------------------
A:node-2>config>system>security# info
----------------------------------------------
...
    user "user1"
        ...
        console
            member "service-22"
        exit
...
```

2.6.3.1 Matching on values in command authorization rules

In addition to matching on command keywords, command authorization profiles allow matching on the values of command elements.

Example: Matching on a command element value (MD-CLI)

```
configure system security aaa local-profiles profile "service-ops" entry 10 match
 "configure service vprn <55>"
```

Example: Matching on a command element value (classic CLI)

```
configure system security profile "service-ops" entry 10 match "configure service vprn
 <55>"
```

The following rules apply to matching on values:

- **Rule 1**
  - **MD-CLI**
    For command authorization on commands that you enter in the MD-CLI, you can specify the match value with or without angle brackets (< >) around the command argument when the **match** is configured in the MD-CLI. If the match value is not in angle brackets when the **match** is configured in the MD-CLI, the entry does not match that value when you enter the command in the classic CLI.
  - **classic CLI**
    For command authorization on commands that you enter in the classic CLI, you must specify the match value in angle brackets in the **match** command argument, as shown in the Matching on a command element value (classic CLI) example.
- **Rule 2**
  - **MD-CLI**
    Only the value of list keys can be matched in a command authorization rule. The value of leafs cannot be matched.
  - **classic CLI**
    The value of any element can be matched in a command authorization rule.
- **Rule 3**

  When a value in angle brackets is present in a **match** command argument, all key values to the left of the value must also be specified (in angle brackets). See Using wildcards in classic CLI command authorization value matching for more information.
- **Rule 4**

  You can either specifically state or completely omit a match value in the match string, as required. However, all unnamed command options in the CLI command must be present in the match string when you use matching on a value. The following example shows how to match on OSPF to allow or deny access to all of OSPF.

  ```
  match "configure router ospf" action deny
  ```

  Alternatively, use the following command to allow a specific OSPF instance for a user.

  ```
  match "configure router ospf <0> <10.10.10.1>" action permit
  ```
- **Rule 5**

  To generate the correct match behavior when multiple unnamed command options are present in the match string, you must provide the command options in the correct order as described in the command help. For example, in the second match example shown in Rule 4, the OSPF instance (0) must come before the router ID (10.10.10.1).

Note: The match behavior may not be achieved if unnamed command options are out of order in the **match** command. Although the user matching is based on the OSPF instance value, that is,

"an unnamed value", all the other unnamed values in the **ospf** command (such as the router ID value) must also be present in the match string.

Example: Match value configuration (MD-CLI)

```
[ex:/configure system security aaa local-profiles profile "default"]
A:admin@node-2# info
    entry 10 {
        match "show router <22> route-table"
        action permit
    }
    entry 20 {
        match "configure service vprn <22>"
        action read-only
    }
    entry 30 {
        match "show service id <22>"
        action permit
    }
    entry 40 {
        match "configure router interface <system>"
        action deny
    }
```

Example: Match value configuration (classic CLI)

```
A:node-2>config>system>security>profile# info
                entry 10
                    match "show router <22> route-table"
                    action permit
                exit
                entry 20
                    match "configure service vprn <22>"
                    action read-only
                exit
                entry 30
                    match "show service id <22>"
                    action permit
                exit
                entry 40
                    match "configure router interface <system>"
                    action deny
                exit
```

2.6.3.2 Using wildcards in classic CLI command authorization value matching

In addition to matching on specific values in command authorization profiles (see Matching on values in command authorization rules), wildcard matching is supported for matching against values in the following commands, when these commands are executed in the classic CLI:

- **oam** commands
- **ping**
- **traceroute**
- **mtrace**

Command authorization using wildcards is only supported in the classic CLI. However, you can configure the wildcard rules in the classic CLI or the MD-CLI.

For example, use the following command for wildcard matching on a value <.*>.

```
match "oam ancp subscriber <.*> loopback"
```

Wildcard value matching is particularly useful when you want to match on a string that contains multiple values, but you only want a specific match on one of the values. The following example shows a command authorization profile with a list of all the permitted IP addresses for ping in router 10.

Example: Using a list of all the permitted IP addresses as match values

```
match ping <10.0.0.1> router <10>
action permit
match ping <10.0.0.2> router <10>
action permit
```

Alternatively, you can use wildcard value matching, which allows a simpler match criterion. In the following example, the use of the <.*> wildcard enables the ability to ping any address in the router 10 context, that is, any address in VRF 10.

Example: Using wildcards for IP address matching

```
match ping <.*> router <10>
action permit
```

Note: While wildcards are available and allowed for all command options in the OAM subtree, Nokia recommends that you exercise caution when using wildcards and limit their use to commands such as **ping**, **traceroute**, and **mtrace**. The use of wildcards in specific formats may be a security concern and result in making the IP addresses in the VRF, including the base routing table, unreachable. Or it could allow the customer to ping any IP address in the VRF, including the base routing table. Avoid this because it is a potential security concern. The following usage, for example, is not advised.

Example: Wildcard usage to avoid

```
match ping <.*> router <.*>
action permit
```

2.6.3.3 CLI session resource management

SR OS has the capability to manage Telnet/SSH sessions per user and at a higher level per system. At the system level, the user can configure a CLI-session group for different customer priorities. The **cli-session-group** command is a container that sets the maximum number of CLI sessions for a class of customers, with a unique session limit for each customer. For example, as shown in Figure 12: Classic CLI session groups for customer classes, "Gold" category customers can have a CLI-session group that allows them more Telnet/SSH sessions compared to "Silver" category customers.

Figure 11: MD-CLI session groups for customer classes

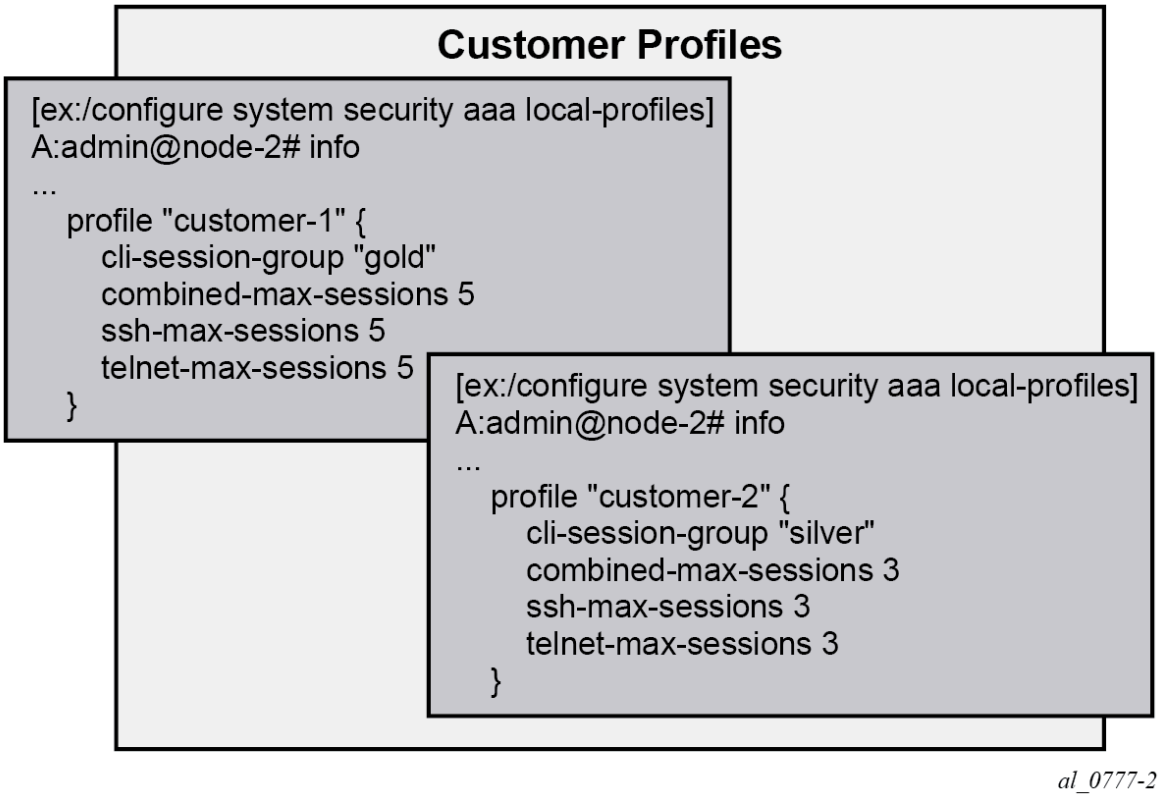
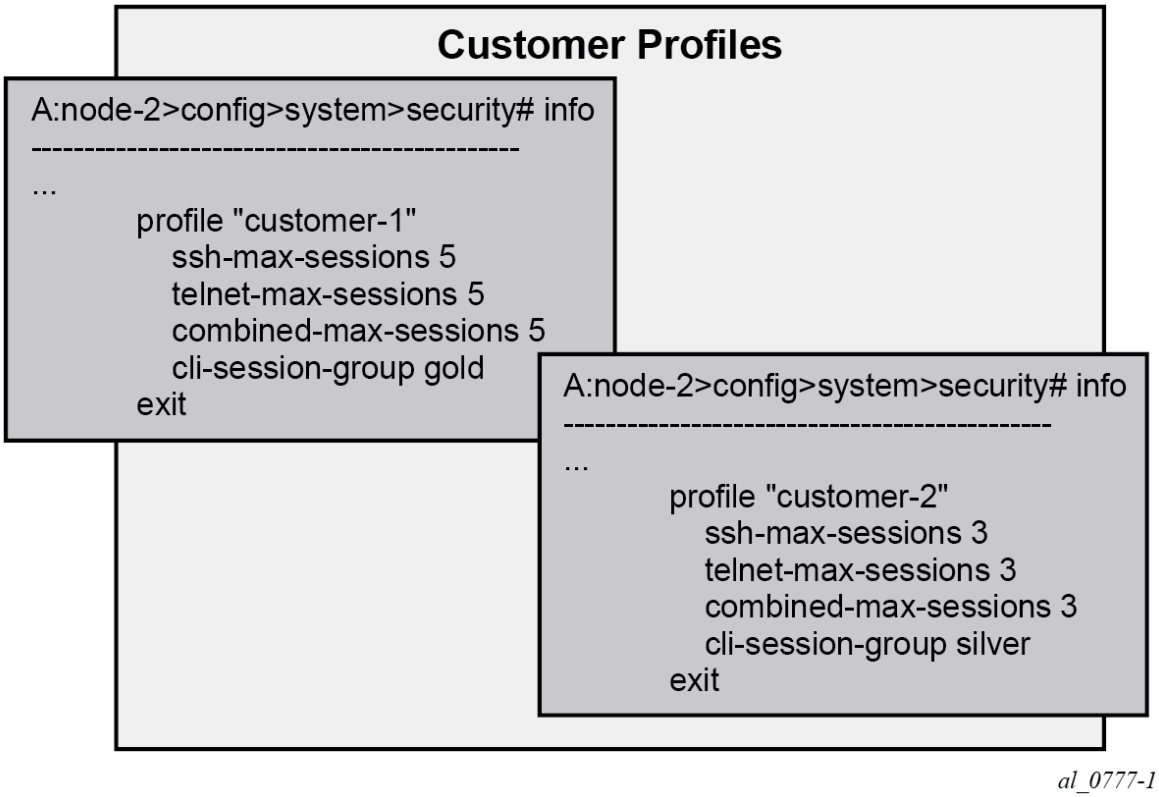


Figure 12: Classic CLI session groups for customer classes



The configured **cli-session-group** can be assigned to user profiles. Each user profile can be configured with its own max SSH/Telnet session and is policed/restricted by the higher order **cli-session-group** that is assigned to it.

As shown in Figure 14: Hierarchy of classic CLI session group profiles, the final picture is a hierarchical configuration with top-level cli-session-groups that control each customer's total number of SSH or Telnet sessions and the user-profile for each user for that customer.

Figure 13: Hierarchy of MD-CLI session group profiles

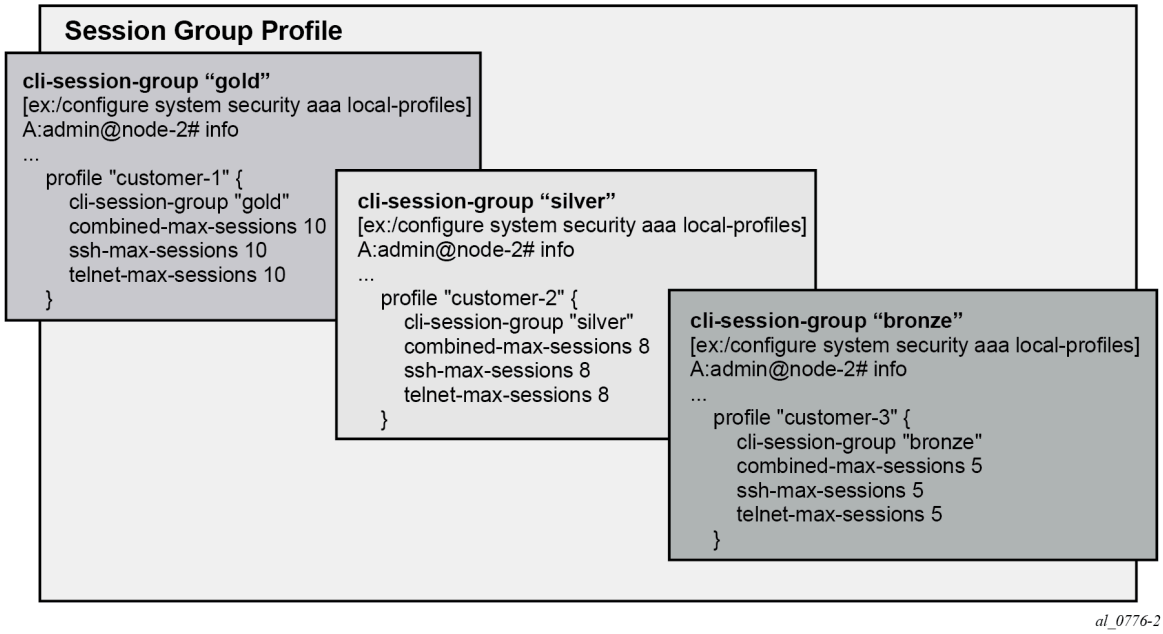
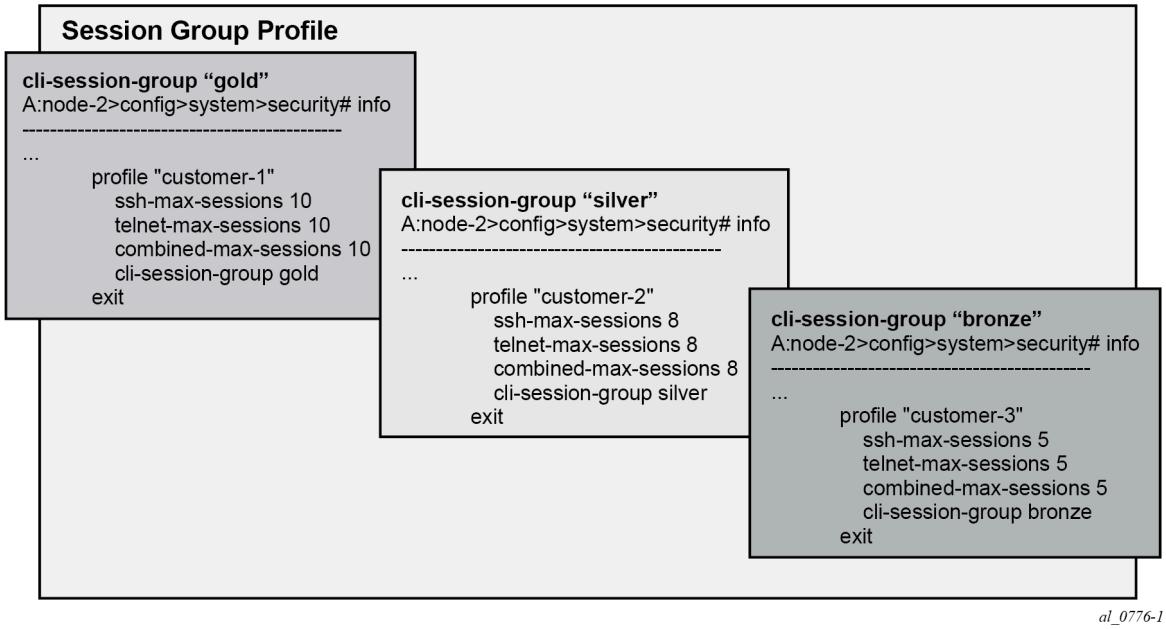


Figure 14: Hierarchy of classic CLI session group profiles



Every profile subtracts one from its corresponding maximum session when a Telnet or SSH session is established in the following cases:

- where multiple profiles are configured under a user

- where multiple profiles arrive from different AAA servers (Local Profile, RADIUS Profile or TACACS Profile)

The first profile to run out of corresponding **max-session** limits future Telnet or SSH sessions. In other words, while each profile for the user can have its independent **max-session**, only the lowest one is honored. If the profile with the lowest **max-session** is removed, the next lower profile **max-session** is honored and so on. All profiles for a user are updated when a Telnet or SSH session is established.

For information about login control, see Configuring login controls.

Use the commands **ssh-max-sessions**, **telnet-max-sessions**, **combined-max-sessions**, and **cli-session-group** in the following context to configure CLI session resources:

- **MD-CLI**

```
configure system security aaa local-profiles profile
```

- **classic CLI**

```
configure system security profile
```

2.6.4 Configuring local users

The username, password, and access permissions are configured for each local user.

Use the following context to configure local users.

- **MD-CLI**

```
configure system security user-params local-user
```

- **classic CLI**

```
configure system security user
```

The following examples provide a sample configuration.

Example: MD-CLI

```
[ex:/configure system security user-params local-user]
A:admin@node-2# info
    user "user1" {
        password "$2y$10$TQrZlpBDra86.qoexZUzQeBXDY1FcdDhGWdD9lLxMuFyPVSm0OGy6"
        restricted-to-home true
        save-when-restricted true
        access {
            console-port-cli true
            ssh-cli true
        }
        console {
            member ["default"]
        }
    }
```

Example: classic CLI

```
A:node-2>config>system>security# info
------------------------------------------------
```

```
...
            user "user1"
                password "$2y$10$TQrZlpBDra86.qoexZUzQeBXDY1FcdDhGWdD9lLxMuFyPVSm0OGy6"
                access console-port-cli ssh-cli
                restricted-to-home
                save-when-restricted
                console
                    member "default"
                exit
            exit
...
-----------------------------------------------
```

2.6.5 Configuring keychain authentication

About this task

The keychain authentication mechanism protects communication between routing protocol neighbors against malicious attacks. Keychain authentication provides the ability to configure authentication keys and update them through key rollover without affecting the state of the routing protocol adjacencies. See Key rollover for more information about **begin-time** and **end-time** configuration.

This procedure describes how to set up keychain authentication.

Note: The user must perform this procedure on both devices that will use keychain authentication to communicate.

Procedure

Step 1. Configure the keychain instance using the following command.

Note: A keychain must be configured on the system before it can be applied to a session.

- **MD-CLI**

```
configure system security keychains keychain
```

- **classic CLI**

```
configure system security keychain
```

Example

Configure a keychain instance (MD-CLI)

```
[ex:/configure system security keychains]
A:admin@node-2# info
    keychain "test1" {
...
```

Example

Configure a keychain instance (classic CLI)

```
A:node-2>config>system>security# info
-----------------------------------------------
...
```

```
            keychain "test1"
                no shutdown
            exit
...
```

Step 2. Use the options under the following context to configure keychain authentication. The keychain must have at least one valid entry and an authentication algorithm.

- **MD-CLI**

```
configure system security keychains keychain
```

- **classic CLI**

```
configure system security keychain
```

Example

Configure keychain authentication (MD-CLI)

```
[ex:/configure system security keychains]
A:admin@node-2# info
    keychain "test1" {
        bidirectional {
            entry 1 {
                authentication-key "LlzaDxIT+KoZkWIOPIhyOILrpoId+Gs= hash2"
                algorithm hmac-sha-1-96
                begin-time 2024-01-01T12:00:00.0+00:00
            }
            entry 2 {
                authentication-key "Yd7MB++GdvjoEHyw4XMTRdHIo0TjRfW+ hash2"
                algorithm hmac-sha-1-96
                begin-time 2014-02-01T12:00:00.0+00:00
            }
        }
    }
```

Example

Configure keychain authentication (classic CLI)

```
A:node-2>config>system>security>keychain# info
----------------------------------------------
                direction
                    bi
                        entry 1 key "LlzaDxIT+KoZkWIOPIhyOILrpoId+Gs=" hash2 algorithm
 hmac-sha-1-96
                            begin-time 2024/01/01 12:00:00 UTC
                        exit
                        entry 2 key "Yd7MB++GdvjoEHyw4XMTRdHIo0TjRfW+" hash2 algorithm
 hmac-sha-1-96
                            begin-time 2014/02/01 12:00:00 UTC
                        exit
                    exit
                exit
                no shutdown
----------------------------------------------
```

Step 3. Associate the configured authentication keychain with a supported protocol.

- **MD-CLI**

Use the **authentication-keychain** command to configure one of the following protocols: BGP, IS-IS, LDP, OSPF, RSVP-TE.

For more information about the **authentication-keychain** command, see the *7705 SAR Gen 2 MD-CLI Command Reference Guide*. For information about configuring the IS-IS **hello-authentication-keychain** command at the interface and interface level contexts, see "Configuring authentication keychain using keychains" in the *7705 SAR Gen 2 Unicast Routing Protocols Guide*.

- **classic CLI**
  Use the **auth-keychain** command to configure one of the following protocols: BGP, IS-IS, LDP, OSPF, RSVP-TE.

  For more information about the **auth-keychain** command, see the *7705 SAR Gen 2 Classic CLI Command Reference Guide*. For information about configuring the IS-IS **hello-auth-keychain** command at the interface and interface level contexts, see the *7705 SAR Gen 2 Unicast Routing Protocols Guide*.

Example

Associate the authentication keychain with a BGP session (MD-CLI)

```
[ex:/configure router "Base" bgp 0]
A:admin@node-2# info
    authentication-keychain "test1"
```

Example

Associate the authentication keychain with a BGP session (classic CLI)

```
A:node-2>config>router>bgp# info
----------------------------------------------
            shutdown
            auth-keychain "test1"
----------------------------------------------
```

2.6.6 Configuring keychains

The following example shows a keychain configuration.

Example: Keychain configuration (MD-CLI)

```
[ex:/configure system security keychains]
A:admin@node-2# info
    keychain "abc" {
        bidirectional {
            entry 1 {
                authentication-key "LHDhK3oGRvkiefQnx7OOc/yutg== hash2"
                algorithm aes-128-cmac-96
                begin-time 2006-12-18T22:55:20.0+00:00
            }
        }
    }
    keychain "basasd" {
        receive {
            entry 1 {
                authentication-key "LHDhK3oGRvkiefQnx7OOc3ib6A== hash2"
                algorithm aes-128-cmac-96
                tolerance infinite
```

```
            }
        }
    }
```

Example: Keychain configuration (classic CLI)

```
A:node-2>config>system>security# info
----------------------------------------------
...
            keychain "abc"
                direction
                    bi
                        entry 1 key "ZcvSElJzJx/wBZ9biCtOVQJ9YZQvVU.S" hash2 alg
orithm aes-128-cmac-96
                            begin-time 2006/12/18 22:55:20
                        exit
                    exit
                exit
            exit
            keychain "basasd"
                direction
                    uni
                        receive
                            entry 1 key "Ee7xdKlYO2DOm7v3IJv/84LIu96R2fZh" hash2
 algorithm aes-128-cmac-96
                                tolerance forever
                            exit
                        exit
                    exit
                exit
            exit
...
----------------------------------------------
```

2.6.7 Copying and overwriting local user and profiles

This section contains information about copying and overwriting a local user or a user profile.

2.6.7.1 Local users

Use the following command to copy a local user to another local user.

- **MD-CLI**

```
configure system security user-params local-user
copy user user-name to user user-name
```

- **classic CLI**

Note: If the destination profile or user already exists, you must specify the **overwrite** option to avoid an error.

```
configure system security copy user user-name to user user-name overwrite
```

The **cannot-change-password** option is not replicated when you perform a **copy user** command. The following example shows the **new-password-at-login** option is created instead.

Example: MD-CLI

```
[ex:/configure system security user-params local-user]
A:admin@node-2# info
...
    }
    user "user1" {
        password "$2y$10$TQrZlpBDra86.qoexZUzQeBXDY1FcdDhGWdD9lLxMuFyPVSm0OGy6"
        access {
            snmp true
        }
        console {
            cannot-change-password true
            member ["default"]
        }
        snmp {
            group "testgroup"
        }
    }
    user "user2" {
        password "$2y$10$TQrZlpBDra86.qoexZUzQeBXDY1FcdDhGWdD9lLxMuFyPVSm0OGy6"
        access {
            snmp true
        }
        console {
            new-password-at-login true
            member ["default"]
        }
        snmp {
            group "testgroup"
        }
    }
...
```

Example: classic CLI

```
A:node-2>config>system>security>user# info
----------------------------------------------
user "user1"
password "$2y$10$TQrZlpBDra86.qoexZUzQeBXDY1FcdDhGWdD9lLxMuFyPVSm0OGy6"
     access snmp
     console
          cannot-change-password
     exit
     snmp
         group "testgroup"
     exit
exit
----------------------------------------------
A:node-2>config>system>security>user# info
----------------------------------------------
user "user2" password "$2y$10$TQrZlpBDra86.qoexZUzQeBXDY1FcdDhGWdD9lLxMuFyPVSm0OGy6"
     access snmp
     console
          new-password-at-login
     exit
     snmp
          group "testgroup"
     exit
exit
----------------------------------------------
```

2.6.7.2 Local user profiles

Use the following command to copy a local user profile to another local user profile:

- **MD-CLI**

```
configure system security aaa local-profiles copy profile user-profile-name to profile user-profile-name
```

- **classic CLI**

```
configure system security copy profile user-profile-name to user-profile-name
```

2.6.8 RADIUS configurations

This section contains information about configuring RADIUS on the SR OS.

2.6.8.1 Configuring RADIUS authentication

RADIUS is disabled by default and must be explicitly enabled. The RADIUS server commands add the RADIUS server. The mandatory configurations to enable the RADIUS server on the local router include the server index, IP address, and secret key. The index determines the sequence in which the servers are queried for authentication requests. In addition, configure the port, and retry and timeout intervals. The other configuration aspects are optional.

Configure the system IP address in order for the RADIUS client to work. For more information, see "Configuring a system interface" in the *7705 SAR Gen 2 Router Configuration Guide*.

Use the commands in the following context to configure RADIUS authentication on a local router:

- **MD-CLI**

```
configure system security aaa remote-servers radius
```

- **classic-CLI**

```
configure system security radius
```

Example: MD-CLI

```
[ex:/configure system security aaa remote-servers radius]
A:admin@node-2# info
    server-retry 5
    server-timeout 5
    server 1 {
        address 10.10.10.103
        secret "G08OmFGXGZjnMgeFRMlrNp8Odc0s hash2"
    }
    server 2 {
        address 10.10.0.1
        secret "YDA64iuZiGG847KPM+7Bvuj3waIn hash2"
    }
    server 3 {
        address 10.10.0.2
```

```
        secret "/WGgOvT3fYcPwh4F5+gGeOVvlfw1 hash2"
    }
    server 4 {
        address 10.10.0.3
        secret "pOYk1obgPtJ2fAq9hcFEJp5tlxKa hash2"
    }
```

Example: classic CLI

```
A:node-2>config>system>security# info
----------------------------------------------
            retry 5
            timeout 5
            server 1 address 10.10.10.103 secret "test1"
            server 2 address 10.10.0.1 secret "test2"
            server 3 address 10.10.0.2 secret "test3"
            server 4 address 10.10.0.3 secret "test4"
...
----------------------------------------
```

2.6.8.2 Configuring RADIUS authorization

For RADIUS authorization to function, enable RADIUS authentication. See Configuring RADIUS authentication.

In addition to the local configuration requirements, configure VSAs on the RADIUS server. See RADIUS VSAs.

Use the following command to configure RADIUS authorization on the local router:

- **MD-CLI**

```
configure system security aaa remote-servers radius authorization
```

- **classic CLI**

```
configure system security radius authorization
```

2.6.8.3 Configuring RADIUS accounting

Use the following command to configure RADIUS accounting on the local router:

- **MD-CLI**

```
configure system security aaa remote-servers radius accounting
```

- **classic CLI**

```
configure system security radius accounting
```

2.6.9 Configuring 802.1x RADIUS policies

Use the commands in the following context to configure generic authentication for clients using 802.1x EAPOL. Additional options are configured per Ethernet port.

See the *7705 SAR Gen 2 Interface Configuration Guide* for more information.

- **MD-CLI**

```
configure system security dot1x radius-policy
```

- **classic CLI**

```
configure system security dot1x radius-plcy
```

The following example shows an 802.1x configuration.

Example: MD-CLI

```
[ex:/configure system security dot1x radius-policy "dot1x_plcy"]
A:admin@node-2# info
    admin-state enable
    source-address 10.1.1.255
    server 1 {
        address 10.1.1.1
        secret "ypeBEsobvcr6wjGzmiPcTfM= hash2"
        authentication-port 65535
    }
    server 2 {
        address 10.1.1.2
        secret "ypeBEsobvcr6wjGzmiPcTXk= hash2"
        authentication-port 65535
    }
```

Example: classic CLI

```
A:node-2>config>system>security# info
----------------------------------------------
            dot1x
                radius-plcy "dot1x_plcy" create
                   server 1 address 10.1.1.1 port 65535 secret "a"
                   server 2 address 10.1.1.2 port 65535 secret "a"
                   source-address 10.1.1.255
                no shutdown
...
----------------------------------------------
```

2.6.10 TACACS+ configurations

This section contains information about configuring TACACS+ on the SR OS.

2.6.10.1 Enabling TACACS+ servers with authentication

To use TACACS+ authentication on the router, configure one or more TACACS+ servers on the network using the following command:

- **MD-CLI**

```
configure system security aaa remote-servers tacplus server
```

- **classic CLI**

```
configure system security tacplus server
```

The following example shows a TACACS+ configuration with authentication.

Example: MD-CLI

```
[ex:/configure system security aaa remote-servers tacplus]
A:admin@node-2# info
    server-timeout 5
    server 1 {
        address 10.10.0.5
        secret "G08OmFGXGZjnMgeFRMlrNn1Ak0Vj hash2"
    }
    server 2 {
        address 10.10.0.6
        secret "YDA64iuZiGG847KPM+7BvsC4f/EL hash2"

    server 3 {
        address 10.10.0.7
        secret "/WGgOvT3fYcPwh4F5+gGeBToxeh5 hash2"
    }
    server 4 {
        address 10.10.0.8
        secret "pOYk1obgPtJ2fAq9hcFEJtCZ/Qj/ hash2"
    }
    server 5 {
        address 10.10.0.9
        secret "oUDAwe2i3vK4MDY7o2KqTdr3cfHp hash2"
    }
```

Example: classic CLI

```
A:node-2>config>system>security>tacplus# info
----------------------------------------------
          timeout 5
          server 1 address 10.10.0.5 secret "test1"
          server 2 address 10.10.0.6 secret "test2"
          server 3 address 10.10.0.7 secret "test3"
          server 4 address 10.10.0.8 secret "test4"
          server 5 address 10.10.0.9 secret "test5"
----------------------------------------------
```

2.6.10.2 Configuring TACACS+ authorization

Enable TACACS+ authentication for TACACS+ authorization to function. See Enabling TACACS+ servers with authentication.

Use the following command to configure TACACS+ authorization on the local router:

- **MD-CLI**

```
configure system security aaa remote-servers tacplus authorization
```

- **classic CLI**

```
configure system security tacplus authorization
```

2.6.10.3 Configuring TACACS+ accounting

Use the following command to configure TACACS+ accounting on the local router:

- **MD-CLI**

```
configure system security aaa remote-servers tacplus accounting
```

- **classic CLI**

```
configure system security tacplus accounting
```

2.6.11 LDAP configurations

2.6.11.1 Configuring LDAP authentication

LDAP is disabled by default and must be explicitly enabled. To use LDAP authentication on the router:

- Configure one or more LDAP servers on the network.
- Ensure that TLS certificates and clients are also configured; see TLS for information about TLS configuration in the SR OS.

Use the commands in the following context to configure LDAP:

- **MD-CLI**

```
configure system security aaa remote-servers ldap
```

- **classic CLI**

```
configure system security ldap
```

The following example shows the LDAP authentication configuration. See LDAP authentication for more information about the use of LDAP authentication in the SR OS.

Example: MD-CLI

```
[ex:/configure system security aaa remote-servers ldap]
A:admin@node-2# info
    admin-state enable
    public-key-authentication true
    server 1 {
        admin-state enable
```

```
        server-name "active-server"
        tls-profile "server-1-profile"
        address 10.1.1.1 {
        }
        bind-authentication {
            root-dn "cn=administrator,cn=users,dc=nacblr2,dc=example,dc=com password"
        }
        search {
            base-dn "dc=sns,dc=example,dc=com"
        }
    }
[ex:/configure system security tls]
A:admin@node-2# info
    client-tls-profile "server-1-profile" {
        admin-state enable
        cipher-list "to-active-server"
        trust-anchor-profile "server-1-ca"
    }
```

Example: classic CLI

```
A:node-2>config>system>security>ldap# info
----------------------------------------------
    public-key-authentication
    server 1 create
        address 10.1.1.1
        bind-authentication "cn=administrator,cn=users,dc=nacblr2,dc=example,dc=com
          password"
        ldap-server "active-server"
        search "dc=sns,dc=example,dc=com"
        tls-profile "server-1-profile"
        no shutdown
    exit
    no shutdown
----------------------------------------------
A:node-2>config>system>security>tls# info
----------------------------------------------
    client-tls-profile "server-1-profile" create
        cipher-list "to-active-server"
        trust-anchor-profile "server-1-ca"
    no shutdown
    exit
```

2.6.11.2 Configuring redundant servers

You can configure up to five redundant LDAP servers. The following examples show configuration of two servers. Server 1 is active and server 5 is backup.

Example: Active server configuration (MD-CLI)

```
[ex:/configure system security aaa remote-servers ldap]
A:admin@node-2# info
    public-key-authentication true
    server 1 {
        server-name "active-server"
        tls-profile "server-1-profile"
        address 10.1.1.1 {
        }
    }
[ex:/configure system security tls]
```

```
A:admin@node-2# info
    client-tls-profile "server-1-profile" {
        admin-state enable
        cert-profile "client-cert-profile"
        cipher-list "to-active-server"
        trust-anchor-profile "server-1-ca"
    }
```

Example: Backup server configuration (MD-CLI)

```
[ex:/configure system security aaa remote-servers ldap]
A:admin@node-2# info
    public-key-authentication true
    server 5 {
        server-name "backup-server-5"
        tls-profile "server-5-profile"
        address 10.5.5.1 {
        }
[ex:/configure system security tls]
A:admin@node-2# info
    client-tls-profile "server-5-profile" {
        admin-state enable
        cert-profile "client-cert-profile"
        cipher-list "to-backup-server 5"
        trust-anchor-profile "server-5-ca"
        }
   }
```

Example: Active server configuration (classic CLI)

```
A:node-2>config>system>security>ldap# info
    public-key-authentication
    server 1 create
        address 10.1.1.1
        ldap-server “active-server”
        tls-profile “server-1-profile”

A:node-2>config>system>security>tls# info
    client-tls-profile “server-1-profile” create
        cert-profile “client-cert-profile”
        cipher-list “to-active-server”
        trust-anchor-profile “server-1-ca”
        no shutdown
    exit
```

Example: Backup server configuration (classic CLI)

```
A:node-2>config>system>security>ldap# info
    public-key-authentication
    server 5 create
        address 10.5.5.1
        ldap-server “backup-server-5”
        tls-profile “server-5-profile”

A:node-2>config>system>security>tls# info
    client-tls-profile “server-5-profile” create
        cert-profile “client-cert-profile”
        cipher-list “to-backup-server-5”
        trust-anchor-profile “server-5-ca”
        no shutdown
    exit
```

2.6.11.3 Configuring login controls

Configure login control for console, Telnet, and FTP sessions. The following example shows a login control configuration.

Example: MD-CLI

```
[ex:/configure system]
A:admin@node-2# info
    login-control {
        exponential-backoff false
        idle-timeout 1440
        motd {
            text "Notice to all users: Software upgrade scheduled 3/2 1:00 AM"
        }
        pre-login-message {
            message "Property of Service Routing Inc. Unauthorized access prohibited."
        }
        ftp {
            inbound-max-sessions 5
        }
        telnet {
            inbound-max-sessions 7
            outbound-max-sessions 2
        }
    }
```

Example: classic CLI

```
A:node-2>config>system# info
----------------------------------------------
...
        login-control
            ftp
                inbound-max-sessions 5
            exit
            telnet
                inbound-max-sessions 7
                outbound-max-sessions 2
            exit
            idle-timeout 1440
            pre-login-message "Property of Service Routing Inc. Unauthorized access
                               prohibited."
            motd text "Notice to all users: Software upgrade scheduled 3/2 1:00 AM"'
        exit
        no exponential-backoff
...
----------------------------------------------
```

3 Model-driven management interfaces

SR OS supports two classes of management interfaces:

- **Classic management interfaces**
  - SNMP
  - the classic CLI
- **Model-driven management interfaces**
  - the MD-CLI (model-driven CLI)
  - NETCONF
  - gRPC (gNMI and gNOI)

Unless otherwise indicated, the term "CLI" in the SR OS user documentation refers to the classic CLI. The classic CLI has been supported in SR OS from the initial introduction of SR OS.

Model-driven management interfaces are based on a common infrastructure that uses YANG models as the core definition for configuration, state, and operational actions. All model-driven interfaces take the same common underlying YANG modules and render them for the particular management interface.

The model-driven interfaces are similar to the classic CLI interfaces with the following notable differences:

- The classic and model-driven configuration formats are incompatible; the system automatically converts the classic configuration to model-driven format when the management interface configuration mode is changed to **model-driven**.
- Some classic CLI branches are moved, renamed, or reorganized in the SR OS YANG modules.
- Many elements use strict references in model-driven interfaces instead of the loose references used in the classic CLI and SNMP.

  See Loose references to IDs and Strict routing policy validation for more information.
- Many elements use string names as keys in model-driven interfaces instead of the numerical identifiers used in the classic CLI and SNMP.

  See String names as keys for more information.
- The classic CLI **shutdown** command is replaced with **admin-state** in model-driven interfaces.
- The classic CLI commands with multiple command options are separated into individual leafs in model-driven interfaces.
- The model-driven interfaces make extensive use of Boolean values (true and false) for configuration settings.
- The default configuration handling is as follows.
  - In model-driven configuration mode, the system operates with "explicit" default handling. Users can set a leaf to the same value as the default and the system displays it as part of the configuration. This handling is similar to RFC 6243 "explicit" mode.
  - In mixed configuration mode, the system uses "explicit" default handling but it is not persistent. Explicitly configured default values are not preserved during a high-availability CPM switchover or a

reboot. Nokia recommends deleting the leaf instead of setting any leaf explicitly to its default value in mixed configuration mode.

- A newly created routing instance, group, or eBGP neighbor in a model-driven interface applies the secure default behavior to reject all routes. Using the **ebgp-default-reject-policy** command to implement this is compliant with RFC 8212. Nokia recommends configuring import and export policies that express the intended routing instead of using the insecure default behavior.

3.1 Management interface configuration modes

The system can operate in different management interface configuration modes, which affects which CLI and network management protocols can be used to configure the system. The following interfaces are available for configuration on SR OS:

- **model-driven (default)** – configuration via model-driven interfaces: the MD-CLI, NETCONF, and gRPC/gNMI, read-only access via the classic CLI and SNMP
- **classic** – configuration via the classic CLI and SNMP, no model-driven interfaces are supported
- **mixed** – configuration via the classic CLI and model-driven interfaces: the MD-CLI, NETCONF, and gRPC/gNMI, read-only access via SNMP

  Mixed configuration mode is useful for operators to migrate from operating in classic management interfaces to a full model-driven mode. It allows the use of previous classic CLI scripts or other OSS integration (for configuration), but with some prerequisites (see Prerequisites for using model-driven management interfaces with classic configurations) and limitations (see the following table).

Use the following command to enable configuration editing by model-driven interfaces.

```
configure system management-interface configuration-mode model-driven
```

Table 20: Management interface configuration mode

		Configuration mode		
		Classic	**Mixed**	**Model-driven**
Classic Interfaces	Classic CLI: configuration write	✓	✓	
	Classic CLI: configuration read	✓	✓	✓
	Classic CLI: non-configuration commands	✓	✓	✓
	SNMP: configuration write	✓		
	SNMP: non-configuration writes (such as admin reboot)	✓		
	SNMP: configuration read	✓	✓	✓
	SNMP: state read	✓	✓	✓
	SNMP: notifications (traps)	✓	✓	✓

		Configuration mode		
		Classic	**Mixed**	**Model-driven**
Model-driven Interfaces with Nokia YANG Models	MD-CLI: configuration write and read		✓	✓
	MD-CLI: state read	✓	✓	✓
	NETCONF: configuration write and read		✓	✓
	NETCONF: state read	✓	✓	✓
	gNMI Set/Get: configuration write and read		✓	✓
	gNMI Get: state read	✓	✓	✓
	gNMI Telemetry: configuration read		✓	✓
	gNMI Telemetry: state read	✓	✓	✓
Saved Configuration File Format	`cf3:\bof.cfg`	Classic	Classic	Classic
	`cf3:\config.cfg`	Classic	Classic	MD
	`cf3:\debug.cfg`	Classic	Classic	MD
	`cf3:\li.cfg`	Classic	Classic	MD
Features	OpenConfig YANG models			✓
	Commit history			✓
	Configuration annotations			✓
	Configuration groups			✓
	MD-CLI **rollback** command			✓
	Classic CLI **admin rollback revert** command	✓	✓	
	Explicit defaults[4]			✓
	Explicit non-deletable SPC objects[5]			✓

4 In model-driven mode, users can set a command option to the same value as the default, and SR OS remembers that it was explicitly set and displays it as part of the configuration. In mixed mode, these values are not persistent and they are lost or forgotten at a CPM high-availability switchover or a reboot.

5 In model-driven mode, users can explicitly create any of the SR OS non-deletable SPC objects, and SR OS remembers that it was explicitly created and displays it as part of the configuration. See System-provisioned configuration objects, for more details about the SPC objects.

		Configuration mode		
		Classic	Mixed	Model-driven
	Configuration changes accepted immediately after a CPM high-availability switchover[6]	✓		✓
	Named route policy entries			✓
	gRPC MD-CLI service for the NISH client		✓	✓
	Remote management using the NISH manager		✓	✓
	MD-CLI command aliases			✓
	Python 3 for **pyexec**, EHS, CRON, and MD-CLI command aliases			✓
	The use of the pySROS library from any location			✓
	Incremental saved configuration files			✓

3.2 YANG data models

Model-driven management interfaces are based on a common infrastructure that uses YANG models as the core definition for configuration, state, and operational actions. All model-driven interfaces (NETCONF, gRPC/gNMI, and the MD-CLI) use the same common underlying YANG modules and render them for the particular management interface. These YANG models are also used for telemetry.

The 7705 SAR Gen 2 supports Nokia YANG data models only.

3.2.1 Nokia SR OS YANG data models

The Nokia SR OS YANG modules are the base for the model-driven architecture.

SR OS configuration is divided into several top level configuration regions (see Datastores and regions). The data models for each configuration region are separated into different YANG modules.

The primary configuration region (**configure**) is modeled in the nokia-conf YANG module specified in a single file located at YANG/nokia-combined/nokia-conf.yang in the SR OS image distribution.

An alternative packaging of the primary configuration region is also available as a set of submodules (for example, nokia-conf-system) that belong to a single module located at YANG/nokia-conf.yang in the SR OS image distribution. The submodules have their own independent revision dates and can be useful to identify which parts of the configuration model have changed.

6 In mixed mode, changes to the configuration are blocked for a few minutes after a CPM high-availability switchover event while the model-driven database is synchronized with the SR OS application layer. There is no impact to running services.

The packaging options (combined and submodule) are alternate representations of the same data model. There is no difference between using the combined or submodule packaging for all the basic configuration or state operations (including with telemetry). The same containers, list, leafs, and so on, exist in the same namespaces whether you are using the combined or submodule packaging. The main difference between the combined and submodule options is seen in the NETCONF <hello>, YANG library, and <get-schema> data where there are lists of modules and submodules.

Some YANG tools may show errors about circular dependencies in the submodules. For example, Pyang gives an error about circular dependencies but does complete the processing to build complete tree or jstree output. If circular dependencies are preventing any necessary tools from correctly processing the YANG, use the combined packaging instead of the submodules. For details about enabling various sets of YANG modules, see the **yang-modules** commands in the *7705 SAR Gen 2 Classic CLI Command Reference Guide*.

The lawful intercept (LI) configuration region is modeled in the nokia-li-conf YANG module specified in a single file called nokia-li-conf.yang.

The BOF configuration region is modeled in the nokia-bof-conf YANG module specified in a single file called nokia-bof-conf.yang.

SR OS state information is modeled in the nokia-state YANG module specified in a single file located at YANG/nokia-combined/nokia-state.yang in the SR OS image distribution.

LI state information is modeled in nokia-li-state.yang which augments the primary nokia-state module.

BOF state information is modeled in nokia-bof-state.yang.

There are also a series of nokia-types-* modules that are included by various configuration and state modules.

The SR OS YANG modules have the following attributes:

- The modules can be used with NETCONF, telemetry, or with the Set/Get RPCs of the gRPC-based gNMI service.
- The modules and submodules indicate the SR OS major release stream using a YANG extension (for example, sros-ext:sros-major-release "rel16";). Module and submodule revisions form a contiguous series of revisions inside a major release stream. There may be two files for the same module with the same revision date but with different contents because they are from two different major release streams. Each active major release stream has revisions ongoing in parallel.

All configuration modules, state modules, and **types** modules are advertised in the SR OS NETCONF server <hello>. Submodules are not advertised in the <hello>.

The common operational **clear**, **show**, **monitor**, and **tools** CLI commands do not have equivalent YANG data models.

Some **admin** and **file** operations have YANG models whereby each operation is modeled using a YANG "action" statement. These can be viewed in the nokia-oper-*.yang files. See YANG-modeled operations for more information.

3.3 Datastores and regions

As described in RFC 8342 a datastore is a conceptual place to store and access information. A datastore maps to an instantiated YANG data tree. See RFC 8342 for more information about datastores.

SR OS supports conventional configuration datastores (for example, running and candidate) as well as some proprietary datastores (for example, li-running).

SR OS also has a proprietary concept called a region (or configuration region). The set of branches and elements in the **configure** branch of the CLI are all located in the primary configuration region simply called **configure**. The majority of SR OS configuration is in the configuration region including ports, interfaces, services and filters. Examples of other regions are:

- **bof** (boot options file)
- **debug** (debugging configuration)
- **li** (lawful intercept)

Each region has its own configuration datastores (running, candidate, and so on). The saved configuration for each region is stored in a separate file on compact flash or remotely (for example, `bof.cfg`, `debug.cfg`, `config.cfg`, `li.cfg`). Regions are independently locked for configuration changes. See the output of the following command in the *7705 SAR Gen 2 Clear, Monitor, Show, Tools CLI Command Reference Guide* for an example of per-region per-datastore information.

```
show system management-interface datastore-locks
```

3.3.1 NMDA support

SR OS supports the Network Management Datastore Architecture (NMDA) for the <intended> and <operational> datastores. When the **nmda-support** command is enabled, the following changes to the YANG model advertisements for NETCONF occur:

- The ietf-yang-library:1.1 revision 2019-01-04 YANG module is advertised in the hello capabilities replacing the ietf-yang-library: 1.0 revision 2016-06-21 version.
- The following additional YANG modules are advertised in the hello capabilities:
  - nokia-datastores
  - ietf-datastores
  - ietf-netconf-nmda
  - ietf-origin
- The ietf-yang-library YANG module revision 2019-01-04 replaces the ietf-yang-library revision 2016-06-21 YANG module when using ietf-netconf-monitoring and ietf-yang-library modules-state.
- The following additional YANG modules are advertised when using ietf-netconf-monitoring and ietf-yang-library modules-state:
  - nokia-datastores
  - ietf-datastores
  - ietf-netconf-nmda
  - ietf-origin

3.4 System-provisioned configuration objects

System-Provisioned Configuration (SPC) objects (configuration list elements and their descendants) are provided as a convenience to users in SR OS.

There are two basic classes of SPC objects: deletable and non-deletable.

Deletable SPC objects are placed into the configuration by SR OS when there is no configuration file at startup time, but can be deleted (removed) by a user. The following characteristics apply to deletable SPC objects:

- Deletable SPC objects can be removed or recreated via NETCONF <edit-config> requests.
- Deletable SPC objects that have not been removed are visible in a NETCONF <get-config> response.
- Deletable SPC objects that have been removed are not visible in model-driven interfaces.
- In the classic CLI these are removed by specifying the keyword **no**, which is then visible in an **info** command or in a saved config (**admin save**); for example, **no log-id 99**.
- The following list summarizes the deletable SPC objects in the SR OS (listed against their MD-CLI paths).

```
configure log filter 1001
configure log log-id (99,100)
configure system security user-params local-user user "admin"
configure system security aaa local-profiles profile "administrative"
configure system security aaa local-profiles profile "default"
configure system security ssh server-cipher-list-v2 cipher (190..230)
configure system security ssh client-cipher-list-v2 cipher (190..230)
configure system security ssh server-mac-list-v2 mac (200..240)
configure system security ssh client-mac-list-v2 mac (200..240)
```

Non-deletable SPC (ND-SPC) objects are not added to the visible configuration by SR OS, but they can be referenced by other parts of the configuration even if they are not visible as part of the configuration. The following characteristics apply to ND-SPC objects:

- Some ND-SPC objects contain leafs (or other descendant elements) that can be modified (for example, **cpu-protection policy 254**). Some ND-SPC objects cannot be modified (for example, **qos sap-ingress "default"**).
- ND-SPC objects are not displayed in model-driven interfaces as part of the configuration unless a user explicitly creates the object. This explicit creation of ND-SPC objects is only supported when operating in **model-driven** configuration mode; it is not supported in **mixed** configuration mode. When a user explicitly creates an ND-SPC object, SR OS remembers that it was explicitly created and displays it as part of the configuration. This may be useful for NETCONF clients and tools that perform offline validation of the configuration against the SR OS YANG models and to resolve leafrefs that point to ND-SPC objects.
- Deleted ND-SPC objects in model-driven interfaces no longer appear as part of the configuration. All descendant elements are reset as unconfigured.
- ND-SPC objects can be referenced by other parts of the configuration regardless of whether they have been modified or created.
- ND-SPC objects created inside a configuration group in model-driven interfaces do not appear in the output of **info intended** or **info inheritance**.
- ND-SPC objects are not displayed in the classic CLI as part of the configuration unless a child or descendant element is modified. Some exceptions to this behavior include the following examples.

  ```
  configure service customer 1 name “1”
  configure system security cpu-protection policy 254
  ```

- ND-SPC objects cannot be deleted in the classic CLI. A deletion attempt returns an error.

- The following non-deletable SPC objects are the common ones used.

```
configure service customer "1"
configure qos sap-egress queue 1
configure qos sap-ingress queue (1,11)
```

- The following list summarizes the non-deletable SPC objects in the SR OS (listed against their MD-CLI paths).

Note: Support for commands in the following list depends on the type of router. See the *7705 SAR Gen 2 MD-CLI Command Reference Guide* for support information.

```
bof port "management"
bof router "management"
bof router interface "management"
configure application-assurance group partition cflowd export-type ("volume","tcp-
performance","rtp-performance","comprehensive","pgw-edr")
configure application-assurance group partition policy app-group "Unknown"
configure application-assurance group partition policy application "Unknown"
configure application-assurance group partition statistics aa-sub-study
 ("protocol","application")
configure call-trace location (cf1,cf2)
configure call-trace trace-profile "default"
configure call-trace trace-profile "default"
configure card fp ingress network pool "default"
configure cflowd sample-profile 1
configure eth-cfm default-domain bridge-identifier <x>
configure filter log 101
configure lag scheduler vlan-qos-policy "default"
configure log log-events (adp event tmnxDiscoveryEndNotify ... wpp event tmnxWppPGHostAuth
Failed)
configure log log-id 101
configure multicast-management bandwidth-policy "default"
configure multicast-management multicast-info-policy "default"
configure multicast-management multicast-info-policy bundle "default"
configure oam-pm bin-group 1
configure oam-pm bin-group bin-type (fd,fdr,ifdv)
configure oam-pm bin-group bin-type bin (0..2)
configure port ethernet egress port-scheduler-policy overrides level (1..8)
configure port ethernet lldp dest-mac (nearest-bridge,nearest-non-tpmr,nearest-customer)
configure port ethernet lldp dest-mac tx-mgmt-address (oob,system,system-ipv6,oob-ipv6)
configure port scheduler vlan-qos-policy "default"
configure qos fp-resource-policy "default"
configure qos fp-resource-policy aggregate-shapers queue-sets size (2..8)
configure qos hw-agg-shaper-scheduler-policy sched-class (1..6)
configure qos network "default"
configure qos network egress fc (be,l2,af,l1,h2,ef,h1,nc)
configure qos network ingress fc (be,l2,af,l1,h2,ef,h1,nc)
configure qos network-queue "default"
configure qos network-queue queue (1,9)
configure qos policer-control-policy root priority-mbs-thresholds priority (1..8)
configure qos policer-control-policy root tier (1,2)
configure qos port-scheduler-policy level (1..8)
configure qos queue-group-templates egress queue-group "policer-output-queues"
configure qos queue-group-templates egress queue-group queue 1
configure qos queue-group-templates egress queue-group sched-class-elevation sched-class
 (1..6)
configure qos sap-egress "default"
configure qos sap-egress queue 1
configure qos sap-egress sched-class-elevation sched-class (1..6)
configure qos sap-ingress "default"
configure qos sap-ingress queue (1,11)
```

```
configure qos scheduler-policy tier (1..3)
configure qos shared-queue "egress-pbr-ingress-queues"
configure qos shared-queue "policer-output-queues"
configure qos shared-queue fc be
configure qos slope-policy "_tmnx_hs_default"
configure qos slope-policy "default"
configure router ("Base","management","vpls-management")
configure router bgp convergence family (ipv4,vpn-ipv4,ipv6,vpn-ipv6,label-ipv4,label-ipv6)
configure router bgp multipath family (ipv4,ipv6,label-ipv4,label-ipv6)
configure router bgp next-hop-resolution labeled-routes transport-tunnel family (vpn,label-
ipv4,label-ipv4)
configure router bgp next-hop-resolution shortcut-tunnel family (ipv4,ipv6)
configure router interface ("system","management")
configure router interface network-domains network-domain "default"
configure router isis 0 igp-shortcut tunnel-next-hop family (ipv4,ipv6,srv4,srv6)
configure router isis interface level (1,2)
configure router isis level (1,2)
configure router isis link-group level (1,2)
configure router isis segment-routing-v6 locator level (1,2)
configure router isis segment-routing-v6 micro-segment-locator level (1,2)
configure router mpls class-forwarding-policy fc (be,l2,af,l1,h2,ef,h1,nc)
configure router mpls interface "system"
configure router mpls lsp auto-bandwidth fc (be,l2,af,l1,h2,ef,h1,nc)
configure router mpls lsp-template auto-bandwidth fc (be,l2,af,l1,h2,ef,h1,nc)
configure router network-domains network-domain "default"
configure router network-domains network-domain “default”
configure router ospf igp-shortcut tunnel-next-hop family (ipv4,srv4)
configure router ospf3 igp-shortcut tunnel-next-hop family (ipv4,ipv6,srv4,srv6)
configure router p2mp-sr-tree p2mp-policy candidate-path path-instances (1,2)
configure router rsvp interface "system"
configure router sgt-qos...
configure service cpipe sap egress qos policer-control-policy overrides root priority-mbs-
thresholds priority (1..8)
configure service cpipe sap ingress qos policer-control-policy overrides root priority-mbs-
thresholds priority (1..8)
configure service customer "1"
configure service epipe sap egress qos policer-control-policy overrides root priority-mbs-
thresholds priority (1..8)
configure service epipe sap ingress qos policer-control-policy overrides root priority-mbs-
thresholds priority (1..8)
configure service ies interface sap egress qos policer-control-policy overrides root
 priority-mbs-thresholds priority (1..8)
configure service ies interface sap ingress qos policer-control-policy overrides root
 priority-mbs-thresholds priority (1..8)
configure service ies subscriber-interface group-interface wlan-gw vlan-range "unmatched"
configure service ipipe sap egress qos policer-control-policy overrides root priority-mbs-
thresholds priority (1..8)
configure service ipipe sap ingress qos policer-control-policy overrides root priority-mbs-
thresholds priority (1..8)
configure service vpls interface sap egress qos policer-control-policy overrides root
 priority-mbs-thresholds priority (1..8)
configure service vpls interface sap ingress qos policer-control-policy overrides root
 priority-mbs-thresholds priority (1..8)
configure service vprn bgp multipath family (ipv4,ipv6,label-ipv4,label-ipv6)
configure service vprn interface sap egress qos policer-control-policy overrides root
 priority-mbs-thresholds priority (1..8)
configure service vprn interface sap ingress qos policer-control-policy overrides root
 priority-mbs-thresholds priority (1..8)
configure service vprn isis interface level (1,2)
configure service vprn isis level (1,2)
configure service vprn isis link-group level (1,2)
configure service vprn sgt-qos...
configure service vprn subscriber-interface group-interface wlan-gw vlan-range "unmatched"
configure subscriber-mgmt gtp peer-profile ("default","default_s11")
```

```
configure subscriber-mgmt ipoe-session-policy "_tmnx_bonding"
configure subscriber-mgmt ipoe-session-policy "default"
configure subscriber-mgmt ppp-policy "default"
configure system alarm-contact-input (1..4)
configure system fp resource-allocation lpm scale-option (1..4)
configure system fp resource-allocation pool (1,2)
configure system power-management 1
configure system ptp router "Base"
configure system security cpu-protection policy (254,255)
configure system security dist-cpu-protection policy ("_default-access-policy","_default-
network-policy","_default-port-policy")
configure system security snmp access group (cli-li-readwrite,cli-readonly,cli-
readwrite,cli-vprn-readwrite,snmp-mgmt,snmp-ro,snmp-rw,snmp-rwa,snmp-trap,snmp-vpls-
mgmt,snmp-vprn,snmp-vprn-ro)
configure system security snmp view (iso,li-view,mgmt-view,no-security,vprn-view)
configure system security user-template {tacplus-default,radius-default,ldap-default}
configure system usb "cf2"
configure test-oam service-activation-testhead acceptance-criteria-template "default"
configure test-oam service-activation-testhead frame-size-template "default"
configure test-oam twamp twamp-light source-udp-port-pools port (64374..64383)
```

3.5 Prerequisites for using model-driven management interfaces with classic configurations

These sections apply to using model-driven management interfaces with existing classic configurations by changing the configuration mode. The classic configuration is automatically converted to model-driven configuration format by the system when the configuration mode is changed to model-driven. Before configuration editing is permitted in model-driven interfaces, **configure system management-interface configuration-mode** must be set to **model-driven** or **mixed** after the prerequisites are completed.

3.5.1 Transitioning between modes

About this task

Perform the following steps before setting the management interface configuration mode to mixed or model-driven.

Procedure

Step 1. Verify that the system configuration only contains commands that are supported in model-driven interfaces. For more information, see "Unsupported configuration in MD interfaces" in the *SR OS R26.x.Rx Software Release Notes*, part number 3HE 29176 000*x* TQZZA.

Step 2. Update the system configuration to meet the prerequisites described in Prerequisites for using model-driven management interfaces with classic configurations.

Step 3. Perform a mode change configuration check as follows.

Use the following command to check if the configuration meets the preceding prerequisite reference requirements to change the management interface configuration mode. Incompatible configuration commands are displayed with an error reason if the prerequisite is not met.

```
tools perform system management-interface configuration-mode check
```

Note: The command does not check if the configuration contains commands that are unsupported in model-driven interfaces. For more information, see section "Unsupported Configuration in MD Interfaces" in the *SR OS R26.x.Rx Software Release Notes*, part number 3HE 29176 000*x* TQZZA.

Example

The following example shows the output of the **configuration-mode check** command when there are incompatible configuration commands.

```
================================================================================
Mode Switch Validation Check
================================================================================
Current Mode       : classic           Desired Mode       : model-driven
Configure          : Errors Detected   LI                 : No Errors
--------------------------------------------------------------------------------
Configuration Validation Errors
--------------------------------------------------------------------------------
1  : MINOR: MGMT_CORE #2004 Incompatible configuration - dynsvc-password
     configured in system security password
2  : MINOR: MGMT_CORE #2004 Incompatible configuration - 'eth-cfm association
     bridge-identifier' reference to service-id exists
3  : MINOR: MGMT_CORE #2004 Incompatible configuration - ca-profile cmpv2 url
     service-id references exist
4  : MINOR: MGMT_CORE #224 Entry does not exist (MD-CLI: configure policy-
     options policy-statement "PEERING_ROUTER_OUT" entry 50 from prefix-list)
--------------------------------------------------------------------------------
Action required: configuration requires updating before mode switch
================================================================================
```

Step 4. Save and back up your configuration. Existing configuration is converted to the MD-CLI format if the mode is changed to model-driven and the saved configuration file is in MD-CLI format.

Step 5. Change the configuration mode to mixed or model-driven as follows.

Note:

- Depending on the size of the system configuration, transitioning from classic mode may take several seconds to several minutes while the model-driven database is populated and synchronized to the current configuration. During the transition period, configuration changes are not allowed and service is not affected.
- Transitioning to classic mode is immediate with no impact to services on the router.

a. If using mixed mode, set the configuration mode to mixed by issuing the following command.

```
configure system management-interface configuration-mode mixed
```

Log out and start a new CLI session to access the MD-CLI engine.

b. If using model-driven mode, set the configuration mode to model-driven by issuing the following command:

```
configure system management-interface configuration-mode model-driven
```

Log out and start a new CLI session to access the MD-CLI engine. When a new user session begins, the MD-CLI engine is available and the MD-CLI prompt is displayed.

Step 6. Save the configuration manually.

- In mixed mode, issue **admin save** from the classic CLI.
- In model-driven mode, issue **admin save** from the MD-CLI.

3.5.2 Configuring the CLI engine

The CLI engine refers to the CLI environment used in a user session (for example, console, Telnet, or SSH) to configure and operate the router. The CLI engine is either the classic CLI engine or the MD-CLI engine, and each can be used in any management interface configuration mode. The following terms are used in this section:

- **preferred CLI engine**

  The CLI engine that is started when a user logs in.

- **authorized CLI engine**

  A CLI engine that a user can switch to (using the CLI engine switch command ("**//**") or where a user can execute commands.

- **active CLI engine**

  The CLI engine that is currently in use for a user session.

The default preferred CLI engine and authorized CLI engines for a session are determined by the management interface configuration mode, which eliminates the need to explicitly configure the CLI engine. With the use of these dynamic defaults, it is possible to transition between the different configuration modes. The following table summarizes the CLI engines for the management interface configuration modes.

Table 21: Management interface configuration modes and CLI engines

Management interface configuration mode	Default preferred CLI engine	Default authorized CLI engines	Allowed CLI engines
classic	classic CLI	classic CLI	MD-CLI (read-only access to configuration), classic CLI
mixed	classic CLI	MD-CLI, classic CLI	MD-CLI, classic CLI
model-driven	MD-CLI	MD-CLI, classic CLI (read-only access to configuration)	MD-CLI, classic CLI (read-only access to configuration)

The preferred and authorized CLI engines can be set globally or per-user to use the classic CLI, the MD-CLI, or both. For changes to the **cli-engine** command to take effect, log out of the CLI session and start a new session. Use the following commands to configured the CLI engine. The first CLI engine configured is the preferred CLI engine.

- **MD-CLI**

```
configure system management-interface cli cli-engine
configure system security user-params local-user user cli-engine
```

- **classic CLI**

```
configure system management-interface cli cli-engine
configure system security user cli-engine
```

The following table summarizes the supported actions for the CLI engine configuration.

Table 22: CLI engine configurations

CLI engine configuration	Preferred CLI engine	Authorized CLI engines	Description
classic-cli	classic CLI	classic CLI	User is restricted to the classic CLI engine
classic-cli md-cli	classic CLI	classic CLI, MD-CLI	User can switch between classic CLI and MD-CLI engines in a session
md-cli classic-cli	MD-CLI	MD-CLI, classic CLI	User can switch between MD-CLI and classic CLI engines in a session
md-cli	MD-CLI	MD-CLI	User is restricted to the MD-CLI engine

3.5.3 Loose references to IDs

A loose reference does not require the target of the reference to exist in the configuration.

For example, when the management interface configuration mode is **classic**, you can configure the following command, even if **ip-filter 37** does not exist in the configuration.

```
configure service pw-template 23 egress filter ip 37
```

Before switching from the **classic** mode to **model-driven** or **mixed**, all loose references using IDs must be replaced with references using string names or removed from the configuration for the following elements:

- all services including the following;

```
configure service vprn
configure service vpls
configure service epipe
```

- the following mirror element

```
configure mirror mirror-dest
```

- the following service elements

```
configure service pw-templates
configure service customer
```

- the following filter elements

```
configure filter ip-filter
configure filter ipv6-filter
configure filter mac-filter
```

- the following QoS elements

```
configure qos network
configure qos sap-ingress
configure qos sap-egress
```

- the following Ethernet CFM elements

```
configure eth-cfm domain
configure eth-cfm association
```

Note: A name can only be assigned to a filter or any element in the preceding list of elements which use IDs as keys in classic interfaces but string names in model-driven interfaces. Nokia recommends assigning names to the elements before an upgrade to Release 15.1.R1.
A name can also be changed in releases before Release 15.1.R1. Elements without names are automatically assigned a name (the ID converted to a string) during an upgrade to Release 15.1.R1 or later, and cannot be changed without manually deleting and recreating the element.

Loose references to IDs for the objects in the preceding list cannot be created while in **mixed** or **model-driven** configuration mode. Any classic CLI scripts must also be updated to avoid the use of any of the following commands.

In the following example, a configuration is shown for the service PW template egress filter.

Example

```
configure service pw-template 23 egress filter ip 37
```

You can change this configuration to the following.

```
configure service pw-template 23 egress filter-name ip ops-sec-filter-a33
```

Because **ip-filter 37** is a loose reference, it does not require a name for the configuration to be valid. However, you may want to assign a name as follows, to make the binding operational.

```
configure filter ip-filter 37 name ops-sec-filter-a33
```

The following lists the set of affected loose references. Some items take a service name as an input. SR OS converts these service names to IDs, and stores the IDs in the configuration. In these cases, the *service-name* becomes an alias at configuration edit time and is not stored as a reference.

IPsec related configuration:

```
configure service vprn interface sap ipsec-tunnel local-gateway-address
configure service vprn interface sap ip-tunnel delivery-service
configure service vprn interface sap l2tpv3-session router
configure service epipe sap l2tpv3-session router
configure service vpls sap l2tpv3-session router
configure service vprn interface sap ipsec-gw default-secure-service
configure service ies interface sap ipsec-gw default-secure-service
```

```
configure service vprn interface sap ipsec-gw dhcp server
configure service ies interface sap ipsec-gw dhcp server
configure service vprn interface sap ipsec-gw dhcp6 server
configure service ies interface sap ipsec-gw dhcp6 server
configure service vprn interface sap ipsec-gw local-address-assignment ipv4 address-
source
configure service vprn interface sap ipsec-gw local-address-assignment ipv6 address-
source
configure service ies interface sap ipsec-gw local-address-assignment ipv4 address-
source
configure service ies interface sap ipsec-gw local-address-assignment ipv6 address-
source
configure service vprn interface sap ipsec-tunnel bfd-enable
configure ipsec client-db client private-service
configure system file-transmission-profile router
```

ETH-CFM, OAM-PM, and SAA:

```
configure eth-cfm domain association bridge-identifier
configure oam-pm session ethernet source mep domain association
configure oam-pm session ip router
configure oam-pm session ip router service-name
configure saa test type cpe-ping service
configure saa test type icmp-ping router
configure saa test type icmp-ping service-name
configure saa test type icmp-trace router
configure saa test type icmp-trace service-name
configure saa test type mac-ping service
configure saa test type mac-trace service
configure saa test type vprn-ping
configure saa test type vprn-ping service
configure saa test type vprn-trace
configure saa test type vprn-trace service
configure saa test type eth-cfm-loopback
configure saa test type eth-cfm-loopback domain
configure saa test type eth-cfm-loopback association
```

Filters:

```
configure service pw-template egress filter ipv6
configure service pw-template egress filter ip
configure service pw-template egress filter mac
configure service pw-template ingress filter ipv6
configure service pw-template ingress filter ip
configure service pw-template ingress filter mac

configure service template epipe-sap-template egress filter ip
configure service template epipe-sap-template egress filter ipv6
configure service template epipe-sap-template egress filter mac
configure service template epipe-sap-template ingress filter ip
configure service template epipe-sap-template ingress filter ipv6
configure service template epipe-sap-template ingress filter mac

configure service template vpls-sap-template egress filter ip
configure service template vpls-sap-template egress filter ipv6
configure service template vpls-sap-template egress filter mac
configure service template vpls-sap-template ingress filter ip
configure service template vpls-sap-template ingress filter ipv6
configure service template vpls-sap-template ingress filter mac

configure li li-filter-block-reservation li-reserved-block ip-filter
configure li li-filter-block-reservation li-reserved-block ipv6-filter
```

```
configure li li-filter-block-reservation li-reserved-block mac-filter
```

PKI:

```
configure system security pki ca-profile cmpv2 url
configure system security pki ca-profile ocsp service
```

QoS:

```
configure service template epipe-sap-template ingress qos
configure service template epipe-sap-template egress qos

configure service template vpls-sap-template ingress qos
configure service template vpls-sap-template egress qos

configure service pw-template ingress qos
configure service pw-template egress qos
```

Subscriber management:

```
configure service ies subscriber-interface group-interface srrp bfd-enable
configure service vprn subscriber-interface group-interface srrp bfd-enable


configure subscriber-mgmt local-user-db ipoe host host-identification service-id
configure subscriber-mgmt local-user-db ipoe host interface service-id
configure subscriber-mgmt local-user-db ipoe host match-radius-proxy-cache server
configure subscriber-mgmt local-user-db ipoe host msap-defaults service
configure subscriber-mgmt local-user-db ipoe host retail-service-id

configure subscriber-mgmt local-user-db ppp host interface service-id
configure subscriber-mgmt local-user-db ppp host l2tp group service-id
configure subscriber-mgmt local-user-db ppp host msap-defaults service
configure subscriber-mgmt local-user-db ppp host retail-service-id

configure subscriber-mgmt msap-policy vpls-only-sap-parameters igmp-snooping mvr
from-vpls

configure service vpls sap msap-defaults service
```

Miscellaneous:

```
configure vrrp policy
configure service vprn interface vrrp bfd-enable

configure service vprn interface ipv6 vrrp bfd-enable
configure router l2tp group ppp default-group-interface service-id
configure router l2tp group tunnel ppp default-group-interface service-id
configure service vprn l2tp group ppp default-group-interface service-id
configure service vprn l2tp group tunnel ppp default-group-interface service-id

configure redundancy multi-chassis peer mc-ring l3-ring in-band-control-path
service-id
configure redundancy multi-chassis peer mc-ring l3-ring ring-node connectivity-
verify service-id
configure redundancy multi-chassis peer mc-ring ring in-band-control-path service-id
configure redundancy multi-chassis peer mc-ring ring ring-node connectivity-verify
service-id

configure open-flow of-switch of-controller vprn
```

3.5.4 Strict routing policy validation

Strict routing policy validation is used for model-driven interfaces. The routing policy must exist for the management interface configuration mode to be changed. Remove references to non-existent routing policies before attempting to switch modes. Strict policy validation is applied to the following routing policy references:

- ARP and ND in the Base router and VPRN instances
- BGP in the Base router and VPRN instances
- global and local variables in main policies and sub-policies
- IGMP, MLD, and PIM in the Base router and VPRN instances
- IS-IS in the Base router and VPRN instances
- LDP
- OSPF and OSPFv3 in the Base router and VPRN instances
- policy-option in **from**, **to**, **action**, and **default-action** statements
- policy-option in sub-policies, **prefix-list**, **as-path**, **as-path-group**, **damping**, and **community** policies
- RIP and RIPng in the Base router and VPRN instances
- RSVP
- single policy-statement or logical policy expressions
- static routes in the Base router and VPRN instances
- subscriber management, except for in **mld-policy** configuration for a local user database (LUDB) host
- VPLS for BGP VSI
- VPRN for GRT, MVPN, and VRF

3.5.5 String names as keys

Many elements use string names as keys in model-driven interfaces instead of the numerical identifiers used in the classic CLI and SNMP.

Note: The string name can only be assigned or modified for these elements in releases before Release 15.1.R1. Elements without names are automatically assigned a name (the identifier converted to a string) during an upgrade to Release 15.1.R1 or later, and cannot be changed without manually deleting and recreating the element.
It is recommended that you assign names to the following elements before an upgrade to Release 15.1 or later:

- all services including the following;

```
configure service vprn
configure service vpls
configure service epipe
```

- the following mirror element

```
configure mirror mirror-dest
```

- the following service elements

```
configure service pw-templates
configure service customer
```

- the following filter elements

```
configure filter ip-filter
configure filter ipv6-filter
configure filter mac-filter
```

- the following QoS elements

```
configure qos network
configure qos sap-ingress
configure qos sap-egress
```

- the following Ethernet CFM elements

```
configure eth-cfm domain
configure eth-cfm association
```

3.6 Commit history

The commit history provides a persistent history of configuration changes committed in model-driven interfaces. A separate history of the last commits (default 50, up to 200) is maintained for each configuration region (bof, configure, debug, and li). Each commit is uniquely identified by a numerical sequential incrementing commit ID assigned by the system.

In the MD-CLI, use the following show command to view the commit history or use the state model:

```
show system management-interface commit-history
```

```
state system management-interface configuration-region commit-history
```

The saved configuration file header also displays the commit history from the last configuration save.

An optional commit comment can be entered using the MD-CLI **commit comment** command or the NETCONF <commit> RPC. Newline separators (\n) can be entered in the comment string to display multiple comment lines.

The following example shows the first commit made by the system when the router boots, followed by two commits by a user with the MD-CLI.

Example: System and user commits with MD-CLI

```
[ex:/configure]
A:admin@node-2# commit comment "Second commit with the MD-CLI."

[ex:/configure]
```

```
A:admin@node-2# commit comment "Third commit with the MD-CLI."

[ex:/configure]
A:admin@node-2# show system management-interface commit-history

===============================================================================
Commit History
===============================================================================
Total Commits : 3

3
  Committed 2022-02-01T11:01:03.8-05:00 by admin (MD-CLI) from 10.1.145.205
  Comment   "Third commit with the MD-CLI."
  Location  "cf3:\config.cfg"
2
  Committed 2022-02-01T11:00:47.7-05:00 by admin (MD-CLI) from 10.1.145.205
  Comment   "Second commit with the MD-CLI."
  Location  "cf3:\config.cfg.1"
1
  Committed 2022-02-01T10:56:01.3-05:00 by system (MD-CLI) from Console
  Log       "System booted version B-22.2.R1."
  Location  "Configuration is not saved to startup."
```

The following example shows a fourth commit made by automation using the NETCONF <commit> RPC with the <comment> augmentation.

Example: NETCONF <commit> RPC with <comment> augmentation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <commit>
        <comment>Fourth commit with NETCONF.</comment>
    </commit>
 </rpc>
]]>]]>
```

Use the following command to display the commit history after the preceding activity.

```
show system management-interface commit-history
```

Output example

```
===============================================================================
Commit History
===============================================================================
Total Commits : 4

4
  Committed 2022-02-01T11:13:38.7-05:00 by admin (NETCONF) from 10.1.236.68
  Comment   "Fourth commit with NETCONF."
  Location  "cf3:\config.cfg"
3
  Committed 2022-02-01T11:01:03.8-05:00 by admin (MD-CLI) from 10.1.145.205
  Comment   "Third commit with the MD-CLI."
  Location  "cf3:\config.cfg.1"
2
  Committed 2022-02-01T11:00:47.7-05:00 by admin (MD-CLI) from 10.1.145.205
  Comment   "Second commit with the MD-CLI."
  Location  "cf3:\config.cfg.2"
1
  Committed 2022-02-01T10:56:01.3-05:00 by system (MD-CLI) from Console
  Log       "System booted version 22.2.R1."
```

```
  Location  "Configuration is not saved to startup."
```

The following usage guidelines apply to the commit history:

- The commit history is supported in model-driven configuration mode only.
- The system files located in the `cf3:\.commit-history` directory must not be edited or deleted, and user files must not be stored there.
- Saved configuration files that are referenced by the commit must not be edited or deleted.
- Editing the BOF from the boot loader does not create a commit history entry.
- Nokia recommends setting the commit history value to at least 50, which is the default value. The commit history can be disabled by setting the value to 0.
- Use the MD-CLI **environment time-format** and **environment time-display** options to change the time formats displayed in the output of the following commands.

  ```
  show system management-interface commit-history
  ```

  ```
  info state system management-interface region-name commit-history
  ```

  ```
  admin show configuration
  ```

  The time formats in **admin show configuration** are in the generated and finished lines.
- The MD-CLI **environment** commands do not change any time formats in the saved configuration file header or footer. These time formats are always written in RFC 3339 format in the Coordinated Universal Time (UTC) or local time zone. Use the following command to configure the value.

  ```
  configure system time prefer-local-time
  ```

3.7 Incremental saved configuration files

When incremental saved configuration files are enabled, the system saves each configuration commit to the **configure** configuration region in a separate incremental saved configuration file, instead of saving a complete saved configuration file each time. This mechanism makes commits over model-driven interfaces (the MD-CLI, NETCONF and gRPC/gNMI) much faster, because less configuration needs to be saved.

When the system boots or the **rollback** command is issued, the last complete saved configuration file is loaded first, and then any required incremental saved configuration files are loaded in the sequence they were committed to apply the previous saved configuration.

The commit history displays information about incremental and complete saved configuration files. The "Location" field displays the complete saved configuration file location, and the "Increment" field displays the incremental saved configuration file location. When the "Location" field is not displayed, the incremental saved configuration file in the "Increment" field is loaded as described above.

Use the following command to show the commit history.

```
show system management-interface commit-history
```

Output example: Commit history showing the incremental saved configuration file location

```
===============================================================================
Commit History
===============================================================================
Total Commits : 2

2
  Committed 2022-06-21T12:55:05.4-04:00 by admin (MD-CLI) from 192.168.0.10
  Increment "cf3:\.commit-history\config-2022-06-21T16-55-05.4Z-4.is"
1
  Committed 2022-06-21T12:49:31.4-04:00 by admin (MD-CLI) from 192.168.0.10
  Increment "cf3:\.commit-history\config-2022-06-21T16-49-31.4Z-3.is"
  Location  "cf3:\config.cfg"
```

A background process generates a complete saved configuration file periodically to reduce the number of incremental saved configuration files that are needed by system. The commit history is updated with a "Location" field like in the following example.

Output example: Commit history showing the complete saved configuration file location

```
===============================================================================
Commit History
===============================================================================
Total Commits : 2

2
  Committed 2022-06-21T12:55:05.4-04:00 by admin (MD-CLI) from 192.168.0.10
  Increment "cf3:\.commit-history\config-2022-06-21T16-55-05.4Z-4.is"
  Location  "cf3:\config.cfg"
1
  Committed 2022-06-21T12:49:31.4-04:00 by admin (MD-CLI) from 192.168.0.10
  Increment "cf3:\.commit-history\config-2022-06-21T16-49-31.4Z-3.is"
  Location  "cf3:\config.cfg.1"
```

Incremental saved configuration files are enabled with the **configure system management-interface configuration-save incremental-saves** command, and must be configured together with the following commands:

- the following command must be **model-driven**

  ```
  configure system management-interface configuration-mode
  ```

- the following commands must be set to **true**:

  ```
  configure system grpc gnmi auto-config-save
  ```

  ```
  configure system management-interface cli md-cli auto-config-save
  ```

  ```
  configure system management-interface netconf auto-config-save
  ```

- the following command must be ≧ 50

  ```
  configure system management-interface commit-history
  ```

- the following command must be ≧ the preceding **commit-history** command

```
configure system management-interface configuration-save configuration-backup
```

The following usage guidelines apply:

- The commit history and incremental saved configuration files in the `cf3:\.commit-history` directory must not be edited by the user
- Multiple configuration save and synchronization events occur because additional system files are saved and synchronized between the active and standby CPM
- The first commit after a system boot or ISSU is followed by a complete save if an **admin save** command was not executed
- The configuration must be saved by executing the **admin save** command before executing the **admin redundancy force-switchover** command

3.8 YANG-modeled operations

In addition to YANG-modeled configuration and state, the SR OS also supports YANG-modeled operations (for example, **admin reboot**, **file remove**).

The SR OS YANG-modeled operations infrastructure applies to MD-CLI and NETCONF interfaces and is supported in any management interface configuration mode (classic, mixed, or model-driven). It is not applicable to operations requested in classic CLI, SNMP, or gRPC interfaces.

YANG-modeled operations are allocated an operation ID. Use the following command to configure the operation ID as an index into the global operations table to examine the details of an operation.

```
state system management-interface operations operation
```

The following operation information can be examined:

- execution status of the operation: in-progress, terminated, or terminated-incomplete
- start-time of the operation
- timeouts associated with the operation

Example: Contents of the global operations table when a file remove-directory command is in progress

```
[/]
A:admin@node-2# info state system management-interface operations
    oldest-operation-id 4
    newest-operation-id 4
    operation 4 {
        asynchronous false
        status in-progress
        start-time 2021-04-13T16:13:18.1+00:00
        request-path "/file/remove-directory"
        session-id 13
        user "admin"
    }
```

Configure and use the operation ID to remove an operation using the following command.

```
admin system management-interface operations delete-operation
```

In the case where the global operations table is full, the **delete-operation** command can optionally be requested with the **op-table-bypass** command option to avoid allocating an *operation-id* and requiring an empty entry in the table.

3.8.1 Asynchronous versus synchronous operations

SR OS supports the following basic response modes for YANG-modeled operations:

- **synchronous**

  This is the default response mode. This mode is supported on MD-CLI and NETCONF.

- **asynchronous**

  This mode is supported only on NETCONF.

In synchronous mode, the response to the operation request contains the complete result data and is held until the operation is complete. No additional operations can be initiated in the same management session (MD-CLI or NETCONF) until the previous operation completes. This behavior is evident in MD-CLI, for example, where the MD-CLI prompt does not return and no input is accepted until the currently running operation is completed.

In asynchronous mode, the response to the operation request does not contain the result data and is sent without waiting for the operation to complete. The request only starts the operation and the client (requester) obtains the result later. Users can perform other commands in the management session while the asynchronous operation runs in the background.

The response to an asynchronous operation request contains an operation ID. This ID is a handle for the operation and allows users to:

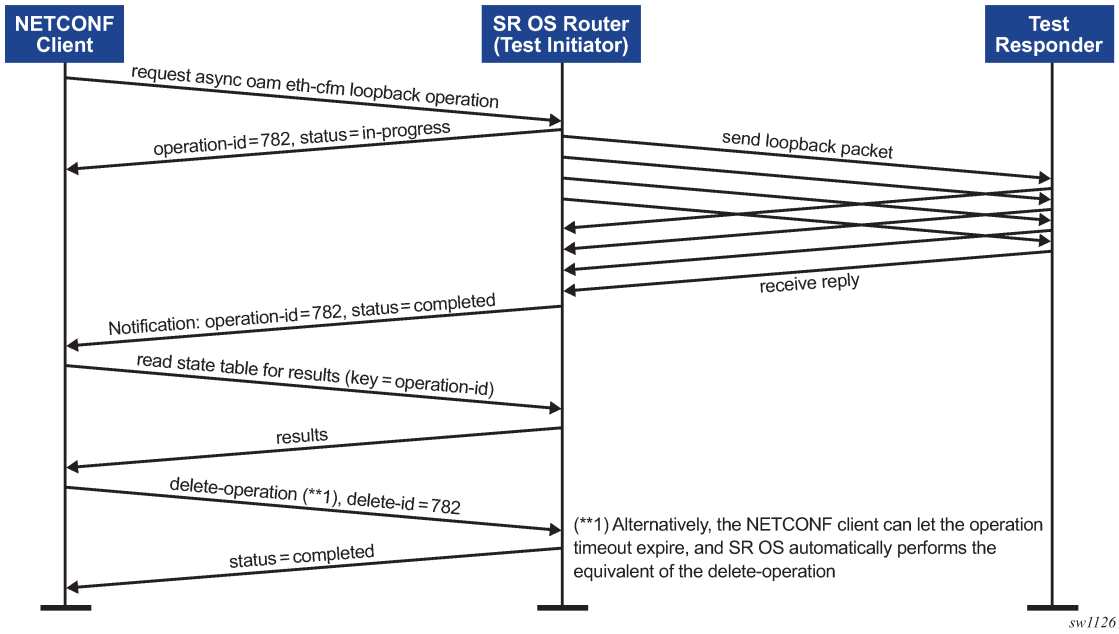
- query the status of the operation
- stop or delete the operation

Synchronous operations require a management session (NETCONF or MD-CLI) for each concurrent operation, whereas a single management session can manage hundreds of concurrent asynchronous operations.

Only a subset of SR OS operational commands are supported in the asynchronous response mode. See the SR OS `nokia-oper-*.yang` files for actions with the "asynchronous" leaf as part of the input to identify operations that support asynchronous mode.

The following figure shows a typical flow for an asynchronous operation.

Figure 15: Asynchronous operation flow



A stopped asynchronous operation (for example, stopped using the **stop-operation** command) stays in the global operations table until it is explicitly deleted using a **delete-operation** command or the retention timeout expires. Synchronous operations are automatically removed from the global operations table when they are completed or stopped.

Note: Because of the parallel processing nature of asynchronous operations, it is possible that an operation completes before the original requester of the operation receives a reply to the request. This means a client could receive a notification about an operation ID that the client does not yet know about.

3.8.2 Examples of operations in MD-CLI

All operations in MD-CLI execute in synchronous response mode.

Example: Operation with no specific result data to return

```
[/]
A:admin@node-2# admin clear security password-history all
```

Example: Operations that returns result data

```
[/]
A:admin@node-2# file version cf3://image/both.tim
TiMOS-C-21.5.R1 for x86_64
Wed May 19 15:02:26 PDT 2021 by builder in /builds/c/215B/R1/panos/main/sros
```

```
[/]
A:admin@node-2# oam eth-cfm loopback aa:bb:cc:dd:ee:22 md-admin-name MyDomain ma-admin-
name MyAssociation mep-id 1 size 0 send-count 5 interval 10 timeout 5
```

```
Eth-Cfm Loopback Test Initiated: Mac-Address: aa:bb:cc:dd:ee:22, out sap: 2/2/1:20

38 bytes; lb_seq=1 passed
38 bytes; lb_seq=2 passed
38 bytes; lb_seq=3 passed
38 bytes; lb_seq=4 passed
38 bytes; lb_seq=5 passed

Sent 5 packets, received 5 packets [0 out-of-order, 0 Bad Msdu]
Packet loss 0.00%
```

3.8.3 Examples of synchronous operations in NETCONF

Example: Synchronous operation that returns no result data

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <admin xmlns="urn:nokia.com:sros:ns:yang:sr:oper-admin">
        <clear>
            <security>
                <password-history>
                    <all/>
                </password-history>
            </security>
        </clear>
    </admin>
  </action>
</rpc>
]]>]]>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
       xmlns:nokiaoper="urn:nokia.com:sros:ns:yang:sr:oper-admin">
    <nokiaoper:operation-id>12</nokiaoper:operation-id>
    <nokiaoper:start-time>2021-06-16T20:11:44.9Z</nokiaoper:start-time>
    <nokiaoper:status>completed</nokiaoper:status>
    <nokiaoper:end-time>2021-06-16T20:11:44.9Z</nokiaoper:end-time>
</rpc-reply>
]]>]]>
```

Example: Synchronous operation that returns result data

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <file xmlns="urn:nokia.com:sros:ns:yang:sr:oper-file">
        <version>
            <url>cf3://image/both.tim</url>
        </version>
    </file>
  </action>
</rpc>
```

```
]]>]]>
```

Response:

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
       xmlns:nokiaoper="urn:nokia.com:sros:ns:yang:sr:oper-file">
    <nokiaoper:operation-id>17</nokiaoper:operation-id>
    <nokiaoper:start-time>2021-06-16T20:37:40.3Z</nokiaoper:start-time>
    <nokiaoper:results>
        <nokiaoper:version>
             <nokiaoper:version-number>C-21.5.R1</nokiaoper:version-number>
             <nokiaoper:version-string>TiMOS-C-21.5.R1 for x86_64 Wed May 19 15:02:26
             PDT 2021 by builder in /builds/c/215B/R1/panos/main/sros</nokiaoper:
             version-string>
        </nokiaoper:version>
    </nokiaoper:results>
    <nokiaoper:status>completed</nokiaoper:status>
    <nokiaoper:end-time>2021-06-16T20:37:40.4Z</nokiaoper:end-time>
</rpc-reply>
]]>]]>
```

To see log events for synchronous operations, MGMT_CORE event #2006 (syncOperationsStatusChange) must first be enabled by configuring it to generate events using the following command:

- **MD-CLI**

```
configure log log-events mgmt-core event syncOperationStatusChange generate
```

- **classic CLI**

```
configure log event-control "mgmt_core" syncOperationStatusChange generate
```

3.8.4 Examples of asynchronous operations in NETCONF

Example: Asynchronous operations in NETCONF

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
    <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <action xmlns="urn:ietf:params:xml:ns:yang:1">
        <global-operations xmlns="urn:nokia.com:sros:ns:yang:sr:oper-global">
          <oam>
            <eth-cfm>
              <loopback>
                <asynchronous>true</asynchronous>
                <destination>aa:bb:cc:dd:ee:22</destination>
                <md-admin-name>MyDomain</md-admin-name>
                <ma-admin-name>MyAssociation</ma-admin-name>
                <mep-id>1</mep-id>
                <send-count>5</send-count>
                <timeout>5</timeout>
                <interval>10</interval>
              </loopback>
            </eth-cfm>
          </oam>
        </global-operations>
      </action>
    </rpc>
```

```
  ]]>]]>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
       xmlns:nokiaoper="urn:nokia.com:sros:ns:yang:sr:oper-global">
    <nokiaoper:operation-id>111</nokiaoper:operation-id>
    <nokiaoper:start-time>2021-06-16T14:17:18.3Z</nokiaoper:start-time>
    <nokiaoper:status>in-progress</nokiaoper:status>
</rpc-reply>
]]>]]>
```

Example: Global operations table status while the operation is running

```
[/]
A:admin@node-2# info state system management-interface operations
    oldest-operation-id 111
    newest-operation-id 111
    operation 111 {
        asynchronous true
        status in-progress
        start-time 2021-06-16T10:17:18.3-04:00
        request-path "/global-operations/oam/eth-cfm/loopback"
        session-id 21
        user "admin"
        execution-timeout {
            time 2021-06-16T11:17:18.3-04:00
            remaining 3599
        }
    }
    next-execution-timeout {
        operation-id 111
        time 2021-06-16T11:17:18.3-04:00
        remaining 3599
    }
```

Use the following command to display log event output when the operation is completed.

```
show log log-id 99
```

Note: Log events for asynchronous actions are enabled by default, but for synchronous operations, the log events are disabled by default. For more information, see Examples of synchronous operations in NETCONF.

The following example shows the log event output.

Example: Log event output when the operation is completed

```
[/]
A:admin@node-2# show log log-id 99

===============================================================================
Event Log 99 log-name 99
===============================================================================
Description : Default System Log
Memory Log contents  [size=500   next event=5  (not wrapped)]

4 2021/06/16 10:17:22.400 EDT WARNING: MGMT_CORE #2005 Base Operation
```

```
"operation-id 111 finished with status completed. Presence of messages in the global
 operations table: error-messages false, warning-messages false, info-messages false."
```

Example: Results available in the state branch

```
[/]
A:admin@node-2# info state eth-cfm domain MyDomain association MyAssociation mep 1
    loopback-results {
        unicast-latest-run {
            test-status completed
            start-time 2021-06-16T10:17:18.0-04:00
            end-time 2021-06-16T10:17:22.0-04:00
            destination-mac-address aa:bb:cc:dd:ee:22
            statistics {
                sent-packets 5
                received-in-order 5
                received-out-of-order 0
                received-bad-msdu 0
                packet-loss 0.0
            }
        }
        multicast-latest-run {
            statistics {
                sent-packets 0
                received-packets 0
            }
        }
    }
```

Example: Delete operation usage to clean up

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <admin xmlns="urn:nokia.com:sros:ns:yang:sr:oper-admin">
        <system>
            <management-interface>
                <operations>
                    <delete-operation>
                       <delete-id>111</delete-id>
                     </delete-operation>
                </operations>
            </management-interface>
        </system>
   </admin>
  </action>
</rpc>
]]>]]>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
       xmlns:nokiaoper="urn:nokia.com:sros:ns:yang:sr:oper-admin">
    <nokiaoper:operation-id>112</nokiaoper:operation-id>
    <nokiaoper:start-time>2021-06-16T14:17:38.5Z</nokiaoper:start-time>
    <nokiaoper:status>completed</nokiaoper:status>
    <nokiaoper:end-time>2021-06-16T14:17:38.6Z</nokiaoper:end-time>
</rpc-reply>
]]>]]>
```

4 SNMP

This chapter provides information to configure the Simple Network Management Protocol (SNMP).

4.1 SNMP overview

This section provides an overview of the Simple Network Management Protocol (SNMP).

4.1.1 SNMP architecture

SNMP is an application-layer protocol that enables communication between managers (the management system) and agents (the network devices). It provides a standard framework to monitor devices in a network from a central location.

An SNMP manager can get a value or set a value via an SNMP agent. The manager uses definitions in the management information base (MIB) to perform operations on the managed device such as retrieving values from variables and processing traps.

The following can occur between the agent and the manager:

- The manager gets information from the agent.
- The manager sets the value of a MIB object in the agent.
- The agent sends traps to notify the manager of significant events that occur on the system.

SNMP can be configured to use UDP (default) or TCP. All PDUs are supported over UDP. The following PDUs are supported over TCP:

- GetBulkRequest
- GetNextRequest
- GetRequest
- Response
- SetRequest

4.1.2 Management information base

A management information base (MIB) is a formal specifications document with definitions of management information used to remotely monitor, configure, and control a managed device or network system. The agent management information of the agent consists of a set of network objects that can be managed with SNMP. Object identifiers are unique object names that are organized in a hierarchical tree structure. The main branches are defined by the Internet Engineering Task Force (IETF). When requested, the IANA assigns a unique branch for use by a private organization or company. The branch assigned to Nokia (TiMetra) is 1.3.6.1.4.1.6527.

The SNMP agent provides management information to support a collection of IETF specified MIBs and a number of MIBs defined to manage devices and network data unique to the Nokia router.

4.1.3 SNMP versions

The agent supports multiple versions of the SNMP protocol:

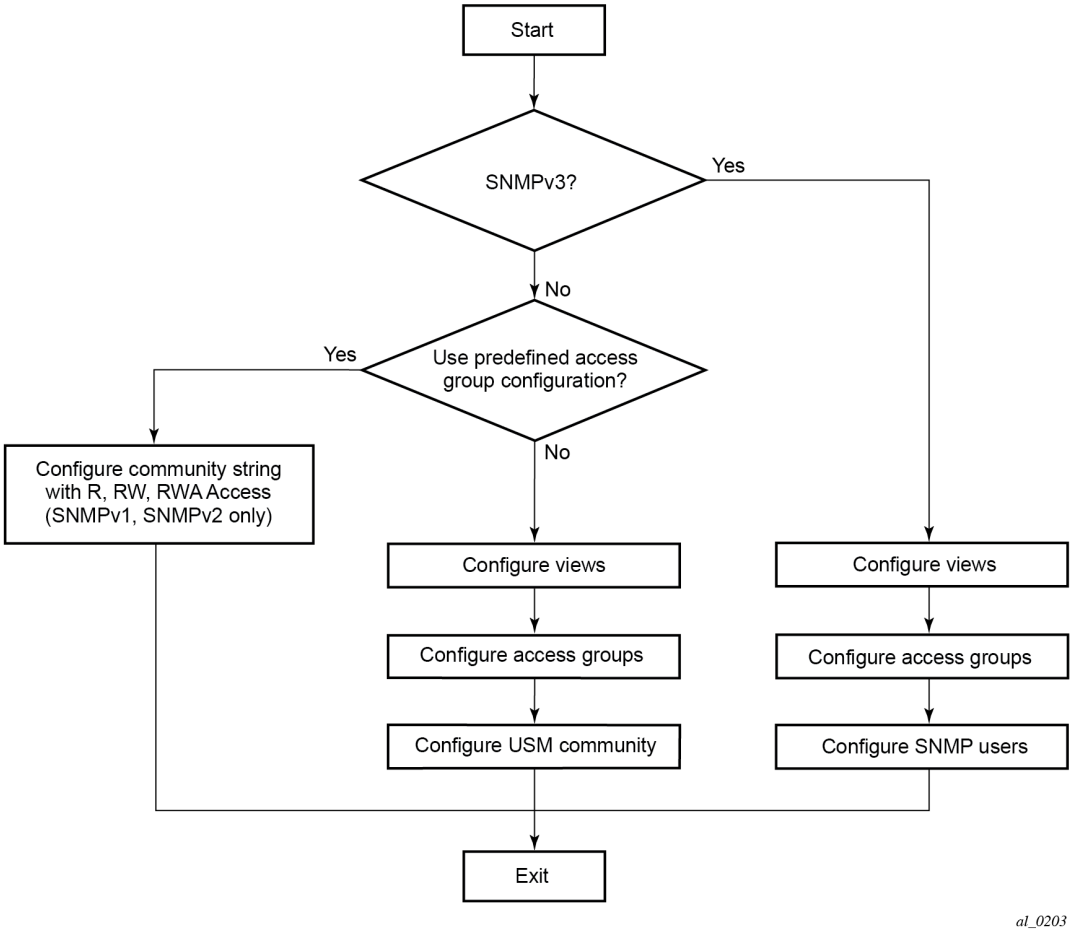
- SNMPv1 is the original Internet-standard network management framework. SNMPv1 uses a community string for authentication.
- SNMPv2c is a community-based administrative framework for SNMPv2. SNMPv2c uses a community string for authentication.
- SNMPv3 uses the User-based Security Model (USM) for user authentication with passwords. View Access Control MIB (VACM) defines the user access control features. The SNMP-COMMUNITY-MIB is used to associate SNMPv1/SNMPv2c community strings with SNMPv3 VACM access control.

SNMPv1 and SNMPv2c do not provide security, authentication, or encryption. Without authentication, a non-authorized user could perform SNMP network management functions and eavesdrop on management information as it passes from system to system. Many SNMPv1 and SNMPv2c implementations are restricted read-only access, which, in turn, reduces the effectiveness of a network monitor in which network control applications cannot be supported.

To implement SNMPv3, an authentication and encryption method must be assigned to a user to be validated by the device. SNMP authentication allows the router to validate the managing node that issued the SNMP message and determine whether the message was tampered with.

The following figure shows the configuration requirements to implement SNMPv1/SNMPv2c, and SNMPv3.

Figure 16: SNMPv1, SNMPv2c, and SNMPv3 configuration and implementation flow



4.1.4 SNMPv3 authentication and privacy protocols

The following SNMPv3 authentication protocols are supported:

- HMAC-MD5-96
- HMAC-SHA-96
- HMAC-SHA-224
- HMAC-SHA-256
- HMAC-SHA-384
- HMAC-SHA-512

The following SNMPv3 privacy protocols are supported:

- CBC-DES
- CFB128-AES-128
- CFB128-AES-192

- CFB128-AES-256

The following combinations of authentication and privacy protocols are not allowed because the hash does not produce enough bytes to use as a key:

- HMAC-MD5-96 (16 bytes) and CFB128-AES-192 (24 bytes)
- HMAC-MD5-96 (16 bytes) and CFB128-AES-256 (32 bytes)
- HMAC-SHA1-96 (20 bytes) and CFB128-AES-192 (24 bytes)
- HMAC-SHA1-96 (20 bytes) and CFB128-AES-256 (32 bytes)
- HMAC-SHA2-224 (28 bytes) and CFB128-AES-256 (32 bytes)

4.1.5 SNMP access control

By default, the SR OS implementation of SNMP uses SNMPv3. SNMPv3 incorporates security model and security level features. A security model is the authentication type for the group and the security level is the permitted level of security within a security model. The combination of the security level and security model determines which security mechanism handles an SNMP packet.

To implement SNMPv1 and SNMPv2c configurations, several access groups are predefined. These access groups provide standard read-only, read-write, and read-write-all access groups and views that can be assigned community strings. To implement SNMP with security features, security models, security levels, and USM communities must be explicitly configured. Optionally, additional views that specify more specific OIDs (MIB objects in the subtree) can be configured.

Access to the information in an SNMPv1/SNMPv2c agent is controlled by the inclusion of a community name string in the SNMP request. The community defines the sub-set of the managed objects of the agent that can be accessed by the requester. It also defines what type of access is allowed: read-only or read-write.

The use of community strings provides minimal security and context checking for both agents and managers that receive requests and initiate trap operations. A community string is a text string that acts like a password to permit access to the agent on the router.

The Nokia implementation of SNMP has defined three levels of community-named access:

- **Read-Only permission**

  This grants read-only access to objects in the MIB, except security objects.

- **Read-Write permission**

  This grants read and write access to all objects in the MIB, except security objects.

- **Read-Write-All permission**

  This grants read and write access to all objects in the MIB, including security objects.

4.1.6 USM community strings

USM community strings associate a community string with an SNMPv3 access group and its view. The access granted with a community string is restricted to the scope of the configured group.

4.1.7 Views

Views control the access to a managed object. The total MIB of a router can be viewed as a hierarchical tree. When a view is created, either the entire tree or a portion of the tree can be specified and made available to a user to manage the objects contained in the subtree. Object identifiers (OIDs) uniquely identify managed objects. Operations are defined for the view such as read, write, or notify.

OIDs are organized in a hierarchical tree with specific values assigned to different organizations. A view defines a subset of the managed objects of the agent, controlled by the access rules associated with that view.

The Nokia SNMP agent associates SNMPv1 and SNMPv2c community strings with a SNMPv3 view.

Use the commands in the following context to configure SNMP views.

```
configure system security snmp view
```

The following system-provisioned views are particularly useful when configuring SNMPv1 and SNMPv2c:

- **iso**

  It is intended for administrative-type access to the entire supported object tree (except Lawful Interception). Use the commands in the following context to configure the "ISO" view automatically associated with any SNMP community that has access permissions of r, re, or raw.

  ```
  configure system security snmp
  ```

- **no security**

  Similar to ISO view, you can remove access to several security areas of the object tree (such as SNMP communities, user and profile configuration, SNMP engine ID, and so on). The no-security view is generally recommended over the ISO view to reduce access to security objects.

- **LI view**

  This view provides access to a small set of Lawful Interception related objects.

- **GMT view**

  This view provides access to IF-MIB and a few other basics. Use the commands in the following context to automatically associate the management view with any SNMP community that has an access-permission of GMT, as configured in the following context.

  ```
  configure system security snmp
  ```

- **VPRN view**

  This limits access to objects associated with a specific VPRN (for example, the per-VPRN logs and SNMP access feature). Use the commands in the following context to automatically associate the VPRN view with any SNMP community configured in the following context.

  ```
  configure service vprn snmp
  ```

4.1.8 Access groups

Access groups associate a user group and a security model to the views the group can access. An access group is defined by a unique combination of a group name, security model (SNMPv1, SNMPv2c, or SNMPv3), and security level (no-authorization-no privacy, authorization-no-privacy, or privacy).

An access group, serves as a template that defines a combination of access privileges and views. A group can be associated with one or more network users to control their access privileges and views.

When configuring access groups, the no-security view is generally recommended over the iso view to restrict access to security objects.

A set of system-provisioned access groups and system-created communities are available in SR OS. The system-provisioned groups and communities that begin with "cli-" are only used for internal CLI management purposes and are not exposed to external SNMP access.

Additional access must be explicitly configured if the preconfigured access groups and views for SNMPv1 and SNMPv2c do not meet security requirements.

4.1.9 Users

By default, authentication and encryption parameters are not configured. Authentication parameters that a user must use to be validated by the device can be modified. SNMP authentication allows the device to validate the managing node that issued the SNMP message and determine whether the message has been tampered with.

User access and authentication privileges must be explicitly configured. In a user configuration, a user is associated with an access group, which is a collection of users who have common access privileges and views (see Access groups).

4.1.10 Per-VPRN logs and SNMP access

Configuration of VPRN-specific logs (with VPRN-specific syslog destinations, SNMP trap, notification groups, and so on) is supported in addition to the global logs configured under **configure log**. By default, the event streams for VPRN logs contain only events that are associated with the particular VPRN.

Use the following command to enable access to the entire system-wide set of events (VPRN and non-VPRN).

```
configure log services-all-events
```

Each VPRN service can be configured with a set of SNMP v1/v2c community strings. These communities are associated with the system provisioned SNMP view called "vprn-view", which limits SNMP access to objects associated with a specific VPRN (along with a few basic system level OIDs).

SNMP communities configured under a VPRN are also associated with the SNMP context "vprn". For example, walking the ifTable (IF-MIB) using the community configured for VPRN 5 returns counters and status for interfaces in VPRN 5 only.

4.2 Best practices for SNMP information retrieval

This section describes best practices for achieving optimal performance when retrieving high volumes of data from SR OS using SNMP.

4.2.1 SNMP GetBulkRequest

Use the SNMP GetBulkRequest method instead of GetRequest or GetNextRequest.

During GetBulkRequest processing, the SR OS SNMP layer uses application data objects from the returned table row to fill in the SNMP reply.

To maximize the advantage of SR OS prefetching and caching optimizations, construct GetBulkRequests with a sequential list of OIDs that represent sequential columns from the same SNMP table row. Enter the objects and OIDs in the GetBulkRequest request to perform a row-by-row retrieval.

Example: GetBulkRequest request row-by-row retrieval

```
interface A, counter 1, counter 2, counter 3, counter N
interface B, counter 1, counter 2, counter 3, counter N
...
interface Z, counter 1, counter 2, counter 3, counter N
```

Do not perform column-by-column retrievals for GetBulkRequest requests, as shown in the following example.

Example: GetBulkRequest request column-by-column retrievals

```
interface A, counter 1
interface B, counter 1
...
interface Z, counter 1
interface A, counter 2
interface B, counter 2
...
interface Z, counter 2
interface A, counter 3
interface B, counter 3
...
interface Z, counter 3
interface A, counter N
interface B, counter N
...
interface Z, counter N
```

To align all responses at the start of a row, avoid performing GetBulkRequests that result in more data than can fit in a single response. This can be accomplished by limiting the max-repetitions depending on the number of repeaters and OIDs, and the size of data returned for each repeater and OID.

4.2.2 Queueing, RTT, and collection performance

The best collection performance is achieved if the SNMP manager keeps the SNMP input queue of the SR OS router filled, but without overflowing it. If maximum performance is required, then the SNMP

manager should always have at least two outstanding requests toward the SR OS router: one that the SR OS router is currently processing (but to which it has not replied yet), and another that is waiting in the SNMP input queue of the SR OS router.

When the SR OS router replies to the request, it immediately processes the next request waiting in the input queue. When the SNMP manager receives the reply, it immediately sends another request to the SR OS SNMP input queue.

If the round trip time (RTT) between the SNMP manager and the SR OS router is significant, the SNMP manager may need to have more than two outstanding requests to maximize collection performance.

The SNMP manager must also avoid sending too many requests at a high rate without waiting for responses. A large number of outstanding requests can cause a backup in the SNMP input queue in SR OS. A backup can cause a long delay in response to the last item in the queue and a timeout on the SNMP manager. It can also cause discards at the SNMP input queue in SR OS.

4.3 Configuration notes

This section describes SNMP configuration caveats.

4.3.1 General

To avoid management systems attempting to manage a partially booted system, SNMP will remain in a shutdown state if the configuration file fails to complete during system startup in classic or mixed configuration mode. While shut down, SNMP gets and sets are not processed. However, notifications are issued if an SNMP trap group has been configured.

To enable SNMP, the portions of the configuration that failed to load must be initialized correctly. Use the following command to start SNMP:

- **MD-CLI**

```
configure system management-interface snmp admin-state enable
```

- **classic CLI**

```
configure system snmp no shutdown
```

Use the following command to change the SNMP engine ID:

Caution: If you change the SNMP engine ID, save the current configuration and reboot the system. This ensures that previously configured SNMP communities and logger trap-target notify communities are valid for the new engine ID.

- **MD-CLI**

```
configure system management-interface snmp engine-id
```

- **classic CLI**

```
configure system snmp engineID
```

4.4 Configuring SNMP with CLI

This section provides information about configuring SNMP with CLI.

4.4.1 SNMP configuration overview

This section describes how to configure SNMP components that apply to SNMPv1 and SNMPv2c, and SNMPv3 on the router.

4.4.1.1 Configuring SNMPv1 and SNMPv2c

Nokia router management is based on SNMPv3. To use the routers with SNMPv1 and/or SNMPv2c, SNMP community strings must be configured. Three predefined access methods are available when SNMPv1 or SNMPv2c access is required. Each access method (**r**, **rw**, or **rwa**) is associated with an SNMPv3 access group that determines the access privileges and the scope of managed objects available. The **community** command is used to associate a community string with a specific access method and the required SNMP version (SNMPv1 or SNMPv2c). The access methods are as follows:

- **Read-Only**

  This grants read only access to the entire management structure with the exception of the security area.
- **Read-Write**

  This grants read and write access to the entire management structure with the exception of the security area.
- **Read-Write-All**

  This grants read and write access to the entire management structure, including security.

If the predefined access groups do not meet your access requirements, then additional access groups and views can be configured. The **usm-community** command is used to associate an access group with an SNMPv1 or SNMPv2c community string. Nokia does not recommend associating a **usm-community** with an SNMP access group that is configured with the **li** (lawful intercept) context.

Use the following command to configure SNMP trap destinations.

```
configure log snmp-trap-group
```

4.4.1.2 Configuring SNMPv3

The SR OS implements SNMPv3. If security features other than the default views are required, the following command options must be configured:

- views
- access groups
- SNMP users

4.4.2 Basic SNMP security configuration

This section provides information to configure SNMP command options and provides examples of common configuration tasks.

At minimum, the following must be configured for SNMP:

- the community string for SNMPv1 and SNMPv2c
- the following for SNMPv3:
  - view
  - SNMP group
  - access
  - SNMP user

The following example shows configuration of SNMP default views, access groups, and access attempts.

Example: MD-CLI

```
[ex:/configure system security snmp]
A:admin@node-2# info
    access "snmp-ro" context "" security-model snmpv1 security-level no-auth-no-privacy {
        read "no-security"
        notify "no-security"
    }
    access "snmp-ro" context "" security-model snmpv2c security-level no-auth-no-privacy {
        read "no-security"
        notify "no-security"
    }
    access "snmp-rw" context "" security-model snmpv1 security-level no-auth-no-privacy {
        read "no-security"
        write "no-security"
        notify "no-security"
    }
    access "snmp-rw" context "" security-model snmpv2c security-level no-auth-no-privacy {
        read "no-security"
        write "no-security"
        notify "no-security"
    }
    access "snmp-rwa" context "" security-model snmpv1 security-level no-auth-no
-privacy {
        read "iso"
        write "iso"
        notify "iso"
    }
    access "snmp-rwa" context "" security-model snmpv2c security-level no-auth-no-privacy
 {
        read "iso"
        write "iso"
        notify "iso"
    }
    access "snmp-trap" context "" security-model snmpv1 security-level no-auth-no-privacy
 {
        notify "iso"
    }
    access "snmp-trap" context "" security-model snmpv2c security-level no-auth-no-privacy
 {
        notify "iso"
    }
```

```
    attempts {
        count 20
        time 5
        lockout 10
    }
    view "iso" subtree "1" {
        mask "ff"
        type included
    }
    view "no-security" subtree "1" {
        mask "ff"
        type included
    }
    view "no-security" subtree "1.3.6.1.6.3" {
        mask "ff"
        type excluded
    }
    view "no-security" subtree "1.3.6.1.6.3.10.2.1" {
        mask "ff"
        type included
    }
    view "no-security" subtree "1.3.6.1.6.3.11.2.1" {
        mask "ff"
        type included
    }
    view "no-security" subtree "1.3.6.1.6.3.15.1.1" {
        mask "ff"
        type included
    }
```

Example: classic CLI

```
A:node-2>config>system>security>snmp# info detail
----------------------------------------------
              view iso subtree 1
                  mask ff type included
              exit
              view no-security subtree 1
                  mask ff type included
              exit
              view no-security subtree 1.3.6.1.6.3
                  mask ff type excluded
              exit
              view no-security subtree 1.3.6.1.6.3.10.2.1
                  mask ff type included
              exit
              view no-security subtree 1.3.6.1.6.3.11.2.1
                  mask ff type included
              exit
              view no-security subtree 1.3.6.1.6.3.15.1.1
                  mask ff type included
              exit
              access group snmp-ro security-model snmpv1 security-level no-auth-no-
privacy read no-security notify no-security
              access group snmp-ro security-model snmpv2c security-level no-auth-no-
privacy read no-security notify no-security
              access group snmp-rw security-model snmpv1 security-level no-auth-no-
privacy read no-security write no-security notify no-security
              access group snmp-rw security-model snmpv2c security-level no-auth-no-
privacy read no-security write no-security notify no-security
              access group snmp-rwa security-model snmpv1 security-level no-auth-no-
privacy read iso write iso notify iso
             access group snmp-rwa security-model snmpv2c security-level no-auth-no-
```

```
privacy read iso write iso notify iso
              access group snmp-trap security-model snmpv1 security-level no-auth-
no-
privacy notify iso
              access group snmp-trap security-model snmpv2c security-level no-auth-
no-privacy notify iso
              attempts 20 time 5 lockout 10
```

4.4.3 Configuring SNMP components

The SNMP components to be configured are the following:

- community string
- view options
- access options
- USM community options
- SNMP command options

4.4.3.1 Configuring a community string

SNMPv1 and SNMPv2c community strings are used to define the relationship between an SNMP manager and agent. The community string acts like a password to permit access to the agent. The access granted with a community string is restricted to the scope of the configured group.

One or more of these characteristics associated with the string can be specified:

- read-only, read-write, and read-write-all permission for the MIB objects accessible to the community
- the SNMP version: SNMPv1 or SNMPv2c

Default access features are preconfigured by the agent for SNMPv1/SNMPv2c.

Use the commands in the following context to configure the community options.

```
configure system security snmp community
```

Example: MD-CLI

```
[ex:/configure system security snmp]
A:admin@node-2# info
    community "IotIpW28Ls8QOTInrJydyerOnvF+U1aq hash2" {
        access-permissions rwa
        version both
    }
    community "X7FnnghnQFm3LicdiQLBGibbOpPGzbdp hash2" {
        access-permissions r
        version v2c
    }
    community "yuumEiY8oD40Uo5/FCkzYi9Uz0Cc2pke hash2" {
        access-permissions r
        version both
    }
```

Example: classic CLI

```
A:node-2>config>system>security>snmp# info
----------------------------------------------
           community "uTdc9j48PBRkxn5DcSjchk" hash2 rwa version both
           community "Lla.RtAyRW2" hash2 r version v2c
           community "r0a159kIOfg" hash2 r version both
----------------------------------------------
```

4.4.3.2 Configuring views

Use the commands in the following context to configure SNMP view options.

```
configure system security snmp view
```

Example: MD-CLI

```
[ex:/configure system security snmp]
A:admin@node-2# info
    view "testview" subtree "1" {
        mask "ff"
    }
    view "testview" subtree "1.3.6.1.2" {
        mask "ff"
        type excluded
    }
```

Example: classic CLI

```
A:node-2>config>system>security>snmp# info
----------------------------------------------
                view "testview" subtree "1"
                    mask ff
                exit
                view "testview" subtree "1.3.6.1.2"
                    mask ff type excluded
                exit

----------------------------------------------
```

4.4.3.3 Configuring access groups

SNMP access group configuration creates an association between an SNMP context, security model, security level, and SNMP views. The access groups can then be used to control SNMP access to the router. Access groups must be configured unless security is limited to the preconfigured access groups and views for SNMPv1 and SNMPv2. The access group is defined by the unique combination of the SNMP context, security model, and security level.

Use the commands in the following context to configure SNMP access groups.

```
configure system security snmp access
```

Example: SNMP access group and view configuration (MD-CLI)

```
[ex:/configure system security snmp]
A:admin@node-2# info
    access "testgroup" context "" security-model usm security-level privacy {
        read "testview"
        write "testview"
        notify "testview"
    }
    view "testview" subtree "1" {
        mask "ff"
    }
    view "testview" subtree "1.3.6.1.2" {
        mask "ff"
        type excluded
    }
```

Example: SNMP access group and view configuration (classic CLI)

```
A:node-2>config>system>security>snmp# info
----------------------------------------------
                view "testview" subtree "1"
                    mask ff
                exit
                view "testview" subtree "1.3.6.1.2"
                    mask ff type excluded
                exit
                access group "testgroup" security-model usm security-level privacy read
 "testview" write "testview" notify "testview"
----------------------------------------------
```

To deploy user-based SNMPv3, combine SNMP access groups with other SNMP configuration options in the configuration of local users. Use the commands in the following context to configure SNMP access and authentication for a user:

- **MD-CLI**

  ```
  configure system security user-params local-user user
  ```

- **classic CLI**

  ```
  configure system security user
  ```

Example: SNMP access and authentication for a user (MD-CLI)

```
[ex:/configure system security user-params local-user user "testuser"]
A:admin@node-2# info
    password "password123"
    access {
        snmp true
        console
    }
    console  {
        member false
    }
     snmp {
        group "testgroup"
        authentication {
            authentication-protocol hmac-md5-96
            authentication-key "9Mx3EjegOfz5g8oZaO49tD/rw2Nlwuvv3H0Uvuppj48= hash2"
```

```
            privacy {
                privacy-protocol cfb128-aes-128
                privacy-key "BkkXpC7xTVuZIsUewNJcuyf8FiOPZJK0oCZ277fRdMY= hash2"
            }
        }
    }
```

Example: SNMP access and authentication for a user (classic CLI)

```
A:node-2>configure system security user "testuser"
A:node-2>config>system>security>user# info
----------------------------------------------
            password "$2y$10$yQPzYQ8B1h.lxUmz3v56M.ekU3S/3V3HNjJ7/4ntrm8B1Oc2S/G/i"
            access snmp
            console
                no member "default"
            exit
            snmp
                authentication hash2 hmac-md5-96 9Mx3EjegOfz5g8oZaO49tD/
rw2Nlwuvv3H0Uvuppj48= privacy cfb128-aes-128 BkkXpC7xTVuZIsUewNJcuyf8FiOPZJK0oCZ277fRdMY=
                group "testgroup"
            exit
----------------------------------------------
```

Note: Use the following command to generate authentication and privacy keys.

```
tools perform system management-interface snmp generate-key
```

An offline tool can also be used to generate the authentication and privacy keys. In addition to the Nokia Network Services Platform (NSP), which includes the **password2key** tool, several third-party tools are also available for use. For example, **snmpv3-hashgen** in the Python SNMPv3-Hash-Generator package generates the correct keys.

4.4.3.4 Configuring USM communities

By default, the SR OS implementation of SNMP uses SNMPv3. However, to implement SNMPv1 and SNMPv2c, USM community strings must be explicitly configured.

USM community strings associate a community string with an SNMPv3 access group and its view. The access granted with a community string is restricted to the scope of the configured group.

Nokia does not recommend associating a **usm-community** with an SNMP access group that is configured with the **li** (lawful intercept) context.

Use the commands in the following context to configure USM community options.

```
configure system security snmp usm-community
```

4.4.3.5 Configuring other SNMP options

Use the commands in the following context to configure SNMP system options, such as an SNMP engine ID that uniquely identifies the node, the maximum SNMP packet size generated by the node, and the port for receiving SNMP request messages and sending replies:

- **MD-CLI**

```
configure system management-interface snmp
```

- **classic CLI**

```
configure system snmp
```

The following example shows configuration of the management interface for SNMP.

Example: MD-CLI

```
[ex:/configure system management-interface snmp]
A:admin@node-2# info
    admin-state disable
    engine-id 0000197f0000daf1ff000000
    general-port 161
    packet-size 1500
```

Example: classic CLI

```
A:node-2>config>system>snmp# info detail
----------------------------------------------
        shutdown
        engineID "0000197f0000daf1ff000000"
        packet-size 1500
        general-port 161
----------------------------------------------\
```

5 NETCONF

This chapter describes the use of the Network Configuration Protocol (NETCONF) SR OS router to perform router management operations.

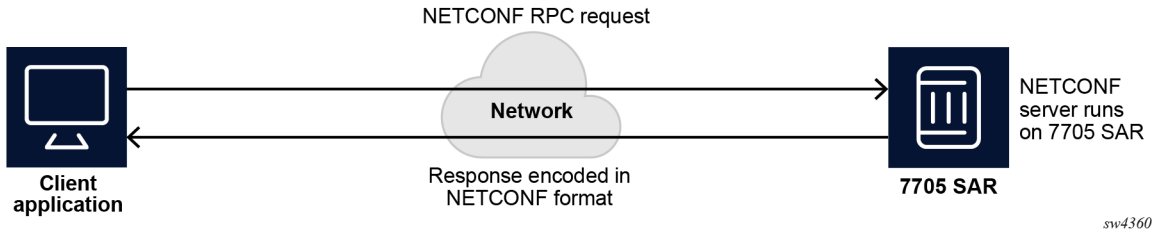
5.1 NETCONF overview

NETCONF is a standardized IETF configuration management protocol specified in RFC 6241, *Network Configuration Protocol (NETCONF)*. It is secure, connection-oriented, and runs on top of the SSHv2 transport protocol as specified in RFC 6242, *Using the NETCONF Configuration Protocol over Secure Shell (SSH)*. NETCONF is an XML-based protocol that can be used as an alternative to CLI or SNMP for managing an SR OS router.

NETCONF uses a Remote Procedure Call (RPC) messaging to facilitate communication between a NETCONF client and the NETCONF server that is running on SR OS. The RPC message and configuration or state data is encoded in an XML document. These XML documents are exchanged between the NETCONF client and a NETCONF server in a series of request and response type of messaging interactions. The SR OS NETCONF interface supports configuration, state, and various router operations (for example, reboot).

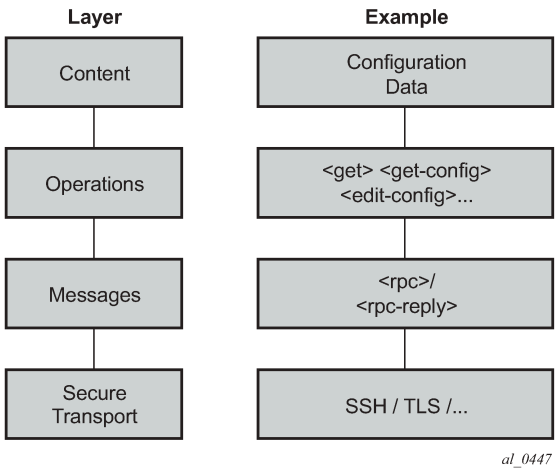
The following figure shows a NETCONF RPC request.

Figure 17: NETCONF RPC request



As defined in RFC 6241, NETCONF can be conceptually partitioned into four layers; these are shown in the following figure.

Figure 18: NETCONF layers (RFC 6241)



5.2 NETCONF in SR OS

The SR OS router can use NETCONF to perform the following router management operations:

- change router configuration using the <edit-config> operation
- read router configuration using the <get-config> operation (equivalent to the **info** command in the SR OS CLI)
- read operational status, data, and associated configuration information using the <get> operation (equivalent to the **show** commands in the SR OS CLI, or executing the info command while in the state branch of MD-CLI)
- send notifications on an SR OS router (equivalent to the SR OS log events)
- operations equivalent to MD-CLI commands, such as **admin**, **file**, **clear**, **oam**, and **ping**, using the md-cli-raw-command request (see NETCONF operations using the md-cli-raw-command request)
- some operations (for example, **admin reboot**, **ping**) are individually modeled as YANG actions and can be called from NETCONF (see Individually YANG-modeled operations)

The SR OS NETCONF server supports both the base:1.1 and the base:1.0 capabilities.

SR OS NETCONF supports both a CLI content layer and an XML-based content layer.

5.2.1 Transport and sessions

SSH transport for NETCONF is supported on a configurable TCP port. The accepted values are port 22, port 830 (default) or any port that is not already in use within the range 1024-49151. NETCONF can be configured to work using IPv4 or IPv6 in-band in the "Base" routing instance or in a VPRN, or out-of-band in the "management" routing instance on the CPM Ethernet ports.

NETCONF SSH sessions (similar to CLI, Secure Copy (SCP), and SSH File Transfer Protocol (sFTP) sessions) are subject to any configurable and non-configurable session limits; for example, inbound-max-

sessions. The SSH server and NETCONF protocol must be enabled in the router configuration to use NETCONF.

Unlike the CLI sessions, NETCONF sessions are not, by default, subject to automatic session timeout. Users may configure an optional **idle-timeout** (in minutes) using the following commands:

- **MD-CLI**

```
configure system management-interface netconf call-home netconf-client idle-timeout
configure system management-interface netconf listen idle-timeout
```

- **classic CLI**

```
configure system netconf call-home netconf-client idle-timeout
configure system netconf listen idle-timeout
```

Users can manually disconnect NETCONF sessions using the **admin disconnect** command.

NETCONF user accounts must exist on the SR OS to enable a client establishing a NETCONF session to log into the router. Use the following command to configure the supported **netconf** access type for the user. For access to the Nokia SR OS YANG data models, only **netconf** access is necessary.

```
configure system security user access
```

Authentication through the local user database is supported for NETCONF users. The **access netconf** statement must be configured in the local user profile. Additionally, because NETCONF runs over SSH, RADIUS/TACACS+ user authentication is also supported.

For RADIUS, **access netconf** must be enabled in the RADIUS default template in the following context:

- **MD-CLI**

```
configure system security aaa user-template user-template-name radius-default
```

- **classic CLI**

```
configure system security user-template radius_default
```

The **use-default-template** command option must be enabled using the following command:

- **MD-CLI**

```
configure system security aaa remote-servers radius use-default-template
```

- **classic CLI**

```
configure system security radius use-default-template
```

If the default template is not used, the RADIUS server must send the Timetra-Access VSA with a value that includes "netconf" access, for example "Timetra-Access = 15".

For TACACS+, **access netconf** must be enabled in the TACACS+ default template in the following context:

- **MD-CLI**

```
configure system security aaa user-template user-template-name tacplus-default
```

- **classic CLI**

```
configure system security user-template tacplus_default
```

The **use-default-template** command option must also be enabled using the following command:

- **MD-CLI**

```
configure system security aaa remote-servers tacplus use-default-template
```

- **classic CLI**

```
configure system security tacplus use-default-template
```

If the TACACS+ default template is not used, the TACACS+ server must send the netconf-access VSA with a value of "true".

Authorization is supported for configuration and state elements in NETCONF. The local, RADIUS, or TACACS+ authorization CLI rules are translated and applied to NETCONF requests to modify or display configuration or state data.

5.2.2 Datastores and URLs

SR OS supports several datastores, including the following standard datastores:

- <running>
- <candidate>
- <startup>
- <intended>
- <operational>

Some NETCONF functions use data from <url> locations.

For additional details about datastores, see Datastores and regions.

The supported datastores can be obtained through the "/yang-library" state data model that contains a list of supported datastores, as defined in RFC 8525.

The following datastores are not available unless support for the NMDA was enabled (see NMDA support):

- <intended>
- <operational>

The :candidate capability is advertised in the SR OS NETCONF server <hello> as:

```
<capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
```

Configuration changes (using the Nokia SR OS YANG data models) made to the <candidate> datastore take effect after a successful <commit> operation.

The <intended> datastore is a read-only representation of the configuration after configuration transformations (such as configuration group expansion) to the <running> datastore are performed.

The <operational> datastore is a read-only representation of the elements the SR OS is using. This includes all configuration (including configuration transformations) and all state information.

The <startup> datastore and <url> can only be used with <copy-config> and <delete-config> and are not supported with any other operations (including <edit-config>, <get-config>, <get>, <validate>, and so on).

The :startup capability is advertised in the SR OS NETCONF server <hello> as:

```
<capability>urn:ietf:params:netconf:capability:startup:1.0</capability>
```

The <url> supports the same options as CLI <file-url>: local urls (CF) and remote urls (FTP and TFTP).

The :url capability is advertised in the SR OS NETCONF server <hello> as:

```
<capability>urn:ietf:params:netconf:capability:url:1.0?scheme=ftp,tftp,file</capability>
```

The following examples show the format of each URL scheme:

- <url>ftp://name:passwd@*IP_ADDRESS*/myfiles/myfile.cfg</url>
- <url>tftp://name:passwd@*IP_ADDRESS*/myfiles/myfile.cfg</url>
- <url>file:///cf3:/myfiles/myfile.cfg</url>
- <url>cf3:/myfiles/myfile.cfg</url>

Note: In the examples, "///" is used to indicate the file URL. Also, the "file://localhost/..." format is not supported.

BOF (Boot Options File) is a separate configuration region, which can be configured over NETCONF using the Nokia SR OS YANG models.

Debug is a separate configuration region, which can be configured over NETCONF using the Nokia SR OS YANG models.

Lawful Intercept (LI) is a separate configuration region, which can be configured over NETCONF using the Nokia SR OS YANG models (including configuring any LI log-ids needed to subscribe to and receive LI NETCONF notifications). The same user permissions apply using NETCONF as with MD-CLI (that is, only LI users can access LI data).

5.2.2.1 Operational datastore

The <operational> datastore is a read-only representation of the elements that the SR OS is using. This includes all configuration (including configuration transformations) and all state information.

The <operational> datastore is only accessible over NETCONF and requires that NMDA support is enabled. To obtain <operational> datastore information, use the <get-data> RPC.

Example: Obtaining information from the <operational> datastore using the <get-data> RPC

```
<rpc message-id="example" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda" xmlns:ds=
"urn:ietf:params:xml:ns:yang:ietf-datastores">
        <datastore>ds:operational</datastore>
    </get-data>
</rpc>
]]>]]>
```

The <get-data> RPC performed on the <operational> datastore can be extended with the **with-origin** option to detail the origin of each configuration node, excluding non-presence containers.

The following rules define the origin:

- YANG non-presence containers do not have an origin.
- The origin for Non-Deletable System Provisioned Objects (for instance, ND-SPC YANG list entries) that were not explicitly configured and for YANG leafs in these list entries is **system**.
- The origin for explicitly configured YANG containers, lists, and leafs (visible in MD-CLI using the **info intended** command) is **intended**.
- The origin for YANG containers, lists, and leafs that are not system-provisioned and not explicitly configured is **default**.

Example: <get-data> RPC usage with the with-origin option

```
<rpc message-id="example" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda" xmlns:ds=
"urn:ietf:params:xml:ns:yang:ietf-datastores">
        <datastore>ds:operational</datastore>
        <with-origin/>
    </get-data>
</rpc>
]]>]]>
```

Specific origins may be selected for output using the mutually exclusive **origin-filter** or **negated-origin-filter** options.

The user can specify the **origin-filter** or **negated-origin-filter** option multiple times in a single **<get-data>** operation.

Example: <get-data> operation using with-origin option and without origin-filter option

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda" xmlns:ds=
"urn:ietf:params:xml:ns:yang:ietf-datastores">
        <datastore>ds:operational</datastore>
        <with-origin/>
        <subtree-filter>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                <system>
                    <name/>
                </system>
            </configure>
            <lldp xmlns="http://openconfig.net/yang/lldp">
                <config>
                    <enabled/>
                </config>
            </lldp>
        </subtree-filter>
    </get-data>
</rpc>
]]>]]>
<rpc-reply message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:or=
"urn:ietf:params:xml:ns:yang:ietf-origin">
            <system>
                <name or:origin="or:intended">testnode</name>
            </system>
        </configure>
        <lldp xmlns="http://openconfig.net/yang/lldp" xmlns:or=
"urn:ietf:params:xml:ns:yang:ietf-origin">
            <config>
```

```
                <enabled or:origin="or:default">true</enabled>
            </config>
        </lldp>
    </data>
</rpc-reply>
]]>]]>
```

Example: <get-data> operation using with-origin option and filtering for default origin

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda" xmlns:ds=
"urn:ietf:params:xml:ns:yang:ietf-datastores">
        <datastore>ds:operational</datastore>
        <with-origin/>
        <subtree-filter>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                <system>
                    <name/>
                </system>
            </configure>
            <lldp xmlns="http://openconfig.net/yang/lldp">
                <config><enabled/></config>
            </lldp>
        </subtree-filter>
        <origin-filter xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">or:default</
origin-filter>
    </get-data>
</rpc>
]]>]]>
<rpc-reply message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
        <lldp xmlns="http://openconfig.net/yang/lldp" xmlns:or=
"urn:ietf:params:xml:ns:yang:ietf-origin">
            <config>
                <enabled or:origin="or:default">true</enabled>
            </config>
        </lldp>
    </data>
</rpc-reply>
]]>]]>
```

Example: <get-data> operation using with-origin option and filtering for default and intended origins

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda" xmlns:ds=
"urn:ietf:params:xml:ns:yang:ietf-datastores">
        <datastore>ds:operational</datastore>
        <with-origin/>
        <subtree-filter>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                <system>
                    <name/>
                </system>
            </configure>
            <lldp xmlns="http://openconfig.net/yang/lldp">
                <config><enabled/></config>
            </lldp>
        </subtree-filter>
        <origin-filter xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">or:default</
origin-filter>
```

```
        <origin-filter xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">or:intended</
origin-filter>
    </get-data>
</rpc>
]]>]]>
<rpc-reply message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:or=
"urn:ietf:params:xml:ns:yang:ietf-origin">
            <system>
                <name or:origin="or:intended">testnode</name>
            </system>
        </configure>
        <lldp xmlns="http://openconfig.net/yang/lldp" xmlns:or=
"urn:ietf:params:xml:ns:yang:ietf-origin">
            <config>
                <enabled or:origin="or:default">true</enabled>
            </config>
        </lldp>
    </data>
</rpc-reply>
]]>]]>
```

5.2.3 NETCONF operations and capabilities

Each RPC request can only contain one operation.

The following table summarizes the protocol operations and capabilities supported on the 7705 SAR Gen 2.

Table 23: Summary of operations and capabilities

Support	Capabilities	Operations
Base protocol operations	—	• <get> • <get-config> • <edit-config> • <copy-config> • <delete-config> • <lock> • <unlock> • <close-session> • <kill-session>
RFC 6241 [7]	writable-running	—
	candidate configuration	• <commit> • <discard-changes>

[7] Optional capabilities defined in RFC 6241 are supported

Support	Capabilities	Operations
	confirmed commit	<cancel-commit>
	validate	<validate>
	startup	—
	URL	—
	rollback-on-error	—
RFC 6243	with-defaults	—
RFC 5277	notification	<create-subscription>
	interleave	—
RFC 6022	ietf-network-monitoring	<get-schema>
RFC 8525	ietf-yang-library	—
RFC 8526	—	<get data>
RFC 7950	—	<action> [8]

The following table lists the supported NETCONF operations and arguments.

Note: Arguments enclosed in square brackets ([ ]) are optional.
X/[configuration-group] means that the [configuration-group] can only be used as a child of *X*.

Table 24: Supported standard NETCONF operations and arguments

Operation	Arguments
get-config	source/[configuration-region] [filter] [format]
edit-config	target/[configuration-region] [default-operation] [test-option] [error-option] config
copy-config	target/[configuration-region] source/[configuration-region]

[8] YANG 1.1 defines an <action> element. See Individually YANG-modeled operations for information about SR OS support for actions.

Operation	Arguments
delete-config	target
lock	target/[configuration-region]
unlock	target/[configuration-region]
get	[filter] [configuration-region] [format]
close-session	—
kill-session	session-id
discard-changes	[configuration-region]
validate	source/[configuration-region]
commit	[confirmed] [confirm-timeout] [comment] [persist] [configuration-region]
cancel-commit	[persist-id]
create-subscription	[stream] [startTime] [stopTime]
get-schema	identifier [version] [format]
get-data	datastore [subtree-filter] [max-depth] [with-defaults] [configuration-region] [format] [with-origin] [config-filter] [origin-filter]

Operation	Arguments
	[negated-origin-filter]
action	See [9]

The following table lists protocol operations and level of support in SR OS NETCONF servers, and limitations, if any, in the current implementation.

Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers

Protocol operation	Example	Supported	Notes

get-config	`<?xml version="1.0" encoding="UTF-8"?>`
	`<rpc message-id="101"`
	`xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">`
	`<get-config>`
	`<source>`
	`<running/>`
	`</source>`
	`</get-config>`
	`</rpc>`
	`]]>]]>`
		Yes	To use the get-config operation, the user must belong to a command-authorization profile with **get-config** enabled under **netconf base-op-authorization**[10].
			If present, configuration annotations are encoded with the nokia-attr:comment attribute.

	`<source>`
	`<startup/>`
	`</source>`
		No	—

	`<source>`
	`<candidate/>`
	`</source>`
		Yes	—

	`<source>` `<config/>`	No	—

9 YANG 1.1 defines an <action> element. See Individually YANG-modeled operations for information about SR OS support for actions.

10 Enable the command in the following context:

- **MD-CLI**

```
configure system security aaa local-profiles profile netconf base-op-authorization
```

- **classic CLI**

```
configure system security profile netconf base-op-authorization
```

Protocol operation	Example	Supported	Notes
	</source>		
	<source> **<url/>** </source>	No	—
	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id=" 101"xmlns="urn:ietf:params:xml:ns:ne tconf:base:1.0"> <get-config> <source><running/></source> **<filter type="subtree">** **</filter>** </get-config> </rpc>]]>]]>	Yes	A <filter> is an optional argument. All subtree filters are supported in SR OS except for "attribute match expressions".
	<filter type="xpath"> **</filter>**	No	—
	<source> **<configuration-region>...</** **configuration-region xmlns=** **"urn:nokia.com:sros:ns:yang:sr:ietf-** **netconf-augments">** </source>	Yes	Optional. Specify "bof", "configure", "debug", or "li". The default, if not specified, is "configure". A datastore must be specified inside the <source> if the <configuration-region> is used.
get	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:ba se:1.0"> <get/> </rpc>]]>]]>	Yes	Retrieves both configuration and state data if in XML content layer. To use the get operation, the user must belong to a command-authorization profile with **get** enabled under **netconf base-op-authorization**[10]. If present, configuration annotations are encoded

Protocol operation	Example	Supported	Notes
			with the nokia-attr:comment attribute.
	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get> **<filter type="subtree"> </filter>** </get> </rpc>]]>]]>	Yes	A <filter> is an optional argument. Subtree filters are supported except for "attribute match expressions".
	<filter type="xpath"> </filter>	No	—
	<configuration-region xmlns= "urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">...</ **configuration-region>**	Yes	Optional. Specify "bof", "configure", "debug", or "li". The default, if not specified, is "configure".
edit-config	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <edit-config> <target/> <default-operation/> <test-option/> <error-option/> <config/> </edit-config> </rpc>]]>]]>	Yes	<default-operation>, <test-option>, and <error-option> are optional arguments. To use the edit-config operation, the user must belong to a command-authorization profile with **edit-config** enabled under **netconf base-op-authorization**[10]. Configuration annotations may be encoded with the nokia-attr:comment attribute to add, change, or delete the annotations.
	<target> **<url/>** </target>	No	—
	<target> **<startup/>**	No	—

Protocol operation	Example	Supported	Notes
	`</target>`		

	`<target>`
	`  <candidate>`
	`</target>`
		Yes	—

	`<default-operation>merge</default-operation>`	Yes	Default
	`<default-operation>none</default-operation>`	Yes	An operation of "none" (inherited or direct) on a leaf node that does not exist in the data model causes SR OS to return an <error-tag> value of "data-missing".
	`<default-operation>replace</default-operation>`	Yes	—
	`<test-option>test-then-set</test-option>`	Yes	—
	`<test-option>set</test-option>`	Yes	—
	`<test-option>test-only</test-option>`	Yes	—
	`<error-option>continue-on-error</error-option>`	No	—
	`<error-option>rollback-on-error</error-option>`	Yes	—
	`<error-option>stop-on-error</error-option>`	Yes	Default. This can be specified but behaves the same as a **rollback-on-error**.
	`<target>`	Yes	Optional.

Protocol operation	Example	Supported	Notes
	<configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">...</configuration-region> </target>		Specify "bof", "configure", "debug", or "li". The default, if not specified, is "configure".
close-session	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <close-session/> </rpc>]]>]]>	Yes	When a session is closed, any locks held are released and the session is terminated. Any pending RPC requests are discarded. To use the close-session operation, the user must belong to a command-authorization profile with **close-session** enabled under **netconf base-op-authorization**[10].
commit	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <commit/> </rpc>]]>]]>	Yes	When commit is issued, any configuration stored in the <candidate> datastore is written to the running configuration unless the device is locked by a NETCONF client or <running> datastore is locked by any other NETCONF session. The startup configuration is also written if **netconf auto-config-save** is configured, in classic CLI in the **configure system** context and in MD-CLI in the **system management-interface** context. To use the commit operation, the user must belong to a command-authorization profile with **commit** enabled under **netconf base-op-authorization**[10].
	<commit>**<comment>Comment entered**	Yes	Optional.

Protocol operation	Example	Supported	Notes
	`over NETCONF.</comment></commit>`		Specifies a comment in the commit history.
	`<commit><confirmed/></commit>`	Yes	Optional Can only be used with the "configure" (default) <configuration-region>.

	`<commit>`
	`<confirmed/>`
	`<confirm-timeout/>`
	`</commit>`
		Yes	Optional
			Can only be used with the "configure" (default) <configuration-region>.

	`<commit>`
	`<confirmed/>`
	`<confirm-timeout/>`
	`<persist/>`
	`</commit>`
		Yes	Optional
			Can only be used with the "configure" (default) <configuration-region>.

	`<commit>`
	`<confirmed/>`
	`<confirm-timeout/>`
	`<persist/>`
	`<persist-id/>`
	`</commit>`
		Yes	Optional
			Can only be used with the "configure" (default) <configuration-region>.

	`<commit>`
	`<configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">...</configuration-region>`
	`</commit>`
		Yes	Optional.
			Specify "bof", "configure", "debug", or "li". The default, if not specified, is "configure".

cancel-commit	`<?xml version="1.0" encoding="UTF-8"?>`
	`<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">`
	`<cancel-commit/>`
	`</rpc>`
	`]]>]]>`
		Yes	Can only be used with the "configure" (default) <configuration-region>
			To use the cancel commit operation, the user must belong to a command-authorization profile with **cancel-commit** enabled under **netconf base-op-authorization**[10].

	`<cancel-commit>` `<persist-id/>...<persist-id/>`	Yes	Optional. When a <commit> <persist> is used, the value of a

Protocol operation	Example	Supported	Notes
	`</cancel-commit>`		<cancel-commit> <persist-id> must be equal to the value used in the <commit> <persist>.

copy-config	`<?xml version="1.0" encoding="UTF-8"?>`
	`<rpc message-id="101"`
	`xmlns:nc=`
	`"urn:ietf:params:xml:ns:netconf:`
	`base:1.0">`
	`  <copy-config>`
	`   <target/>`
	`   <source/>`
	`  </copy-config>`
	`</rpc>`
	`]]>]]>`
		Yes	The <copy-config> operation is supported for specific combinations of source and target datastores.
			To use the copy-config operation, the user must belong to a command-authorization profile with **copy-config** enabled under **netconf base-op-authorization**[10].
			When the specified <configuration-region> is "li", the <source><running/> to <target><startup/> becomes the only valid <copy-config> combination.

	`<target><running/></target>`	No	The running datastore cannot be a <target> for a <copy-config>.
	`<source><running/></source>` `<target><candidate/></target>`	No	Use <discard-changes>.
	`<source><running/></source>` `<target><startup/></target>`	Yes	Equivalent to **admin save.** If NETCONF auto-config-save is not enabled, this should be issued after every <commit> for the change to be persistent over a reboot. An index file is also saved in classic management-interface configuration-mode if **persist on** is configured in the bof.
	`<source><running/></source>` `<target><url/></target>`	Yes	Equivalent of **admin save** *file-url*.

Protocol operation	Example	Supported	Notes
			An index file is also saved in classic management-interface configuration-mode if **persist on** is configured in the bof.
	`<source><startup/></source>` `<target><candidate/></target>`	Yes	—
	`<source><startup/></source>` `<target><url/></target>`	Yes	Supported if both source and target are not remote URLs. Only configuration changes are saved; for example, an index file is not saved in classic management-interface configuration-mode even if **persist on** is configured in the bof.
	`<source><startup/></source>` `<target><running/></target>`	No	—
	`<source><candidate/></source>` `<target><startup/></target>`	Yes	—
	`<source><candidate/></source>` `<target><url/></target>`	Yes	—
	`<source><candidate/></source>` `<target><running/></target>`	No	Use <commit> instead.
	`<source><url/></source>` `<target><running/></target>`	No	—
	`<source><url/></source>` `<target><candidate/></target>`	Yes	Supported in model-driven configuration only.
	`<source><url/></source>` `<target><startup/></target>`	Yes	Supported if both source and target are not remote URLs. Only configuration changes are saved; for example,

Protocol operation	Example	Supported	Notes
			an index file is not saved in classic management-interface configuration-mode even if **persist on** is configured in the BOF.
	`<source><url/></source>` `<target><url/></target>`	Yes	Supported if both source and target are not remote URLs.
	`<target><config/></target>`	No	—
	`<source><config/></source>` `<target><candidate/></target>`	Yes	—
	`<source><config/></source>` `<target><startup/></target>`	Yes	—
	`<source><config/></source>` `<target><url/></target>`	Yes	—

	`<target>`
	`  <configuration-region xmlns=`
	`"urn:nokia.com:sros:ns:yang:sr:ietf-`
	`netconf-augments">`
	`  ...`
	`  </configuration-region>`
	`</target>`
		Yes	Optional.
			Specify "bof", "configure", "debug", or "li". The default, if not specified, is "configure".
			Not supported in mixed mode.

	`<source>`
	`  <configuration-region xmlns=`
	`"urn:nokia.com:sros:ns:yang:sr:ietf-`
	`netconf-augments">`
	`  ...`
	`  </configuration-region>`
	`</source>`
		Yes	Optional.
			Specify "bof", "configure", "debug", or "li". The default, if not specified, is "configure".
			Not supported in mixed mode.

kill-session	`<?xml version="1.0" encoding="UTF-8"?>`
	`<rpc message-id="101"`
	`xmlns="urn:ietf:params:xml:ns:netconf:ba`
	`se:1.0">`
	`  <kill-session>`
	`  **<session-id/>**`
	`  </kill-session>`
	`</rpc>`
		Yes	A NETCONF session cannot kill itself.
			A NETCONF session cannot kill a non-NETCONF session.
			When a session is killed, any operations pending in

Protocol operation	Example	Supported	Notes
	]]>]]>		that session are discarded. Any locks held by that session are released. Only a NETCONF user that belongs to a command-authorization profile with **kill-session** enabled under **netconf base-op-authorization**[10] can kill a NETCONF session.
lock	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <lock> <target/> </lock> </rpc>]]>]]>	Yes	Only a NETCONF user that belongs to a command-authorization profile with **netconf base-op-authorization lock** enabled, can lock a datastore.[10]
	<target><**candidate/**></target>	Yes	Locking the <candidate> datastore implicitly locks both the <running> and <candidate> datastores.
	<target><**running/**></target>	Yes	Locking the <running> datastore locks both the <running> and <candidate> datastores.
	<target><**startup/**></target>	No	—
	<target><**url/**></target>	No	—
	<target> <**configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">...</configuration-region**> </target>	Yes	Optional. Specify "bof", "configure", "debug", or "li". The default, if not specified, is "configure".
unlock	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <unlock> <**target/**>	Yes	Only a NETCONF user that belongs to a command-authorization profile with **lock** enabled under **netconf base-**

Protocol operation	Example	Supported	Notes
	</unlock> </rpc>]]>]]>		**op-authorization**[10] can unlock a datastore. (**lock** authorization also gives unlock authorization.) A datastore lock is unlocked when: • using <unlock> • disconnecting a NETCONF session (from CLI using the **admin disconnect** command, by using **Ctrl-c**, by performing <kill-session>, or by performing <close-session>) Upon unlocking/ disconnecting a NETCONF session that had acquired a datastore lock, SR OS: • releases the lock • discards any "uncommitted" changes
	<target><**candidate/**></target>	Yes	—
	<target><**running/**></target>	Yes	—
	<target><**startup/**></target>	No	—
	<target><**url/**></target>	No	—
	<target> <configuration-region xmlns= "urn:nokia.com:sros:ns:yang:sr:ietf- netconf-augments"> ... </configuration-region> </target>	Yes	Optional. Specify "bof", "configure", "debug", or "li". The default, if not specified, is "configure".
validate	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id="101"	Yes	XML content layer only.

Protocol operation	Example	Supported	Notes
	xmlns:nc= "urn:ietf:params:xml:ns:netconf: base:1.0"> <validate> <source/> </validate> </rpc>]]>]]>		Only syntax validation is performed. If more than one error exists, SR OS returns multiple errors. No semantic validation is performed. To use the validate operation, the user must belong to a command-authorization profile with **validate** enabled under **netconf base-op-authorization**[10].
	<source><**candidate/**></source>	Yes	—
	<source><**running/**></source>	No	—
	<source><**startup/**></source>	No	—
	<source><**url/**></source>	No	—
	<source><**config/**></source>	No	—
	<source> <configuration-region xmlns= "urn:nokia.com:sros:ns:yang:sr:ietf- netconf-augments"> ... </configuration-region> </source>	Yes	Optional. Specify "bof", "configure", "debug", or "li". The default, if not specified, is "configure".
delete-config	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id="101" xmlns:nc= "urn:ietf:params:xml:ns:netconf: base:1.0"> <delete-config> <target/> </delete-config> </rpc>	Yes	To use the delete-config operation, the user must belong to a command-authorization profile with **delete-config** enabled under **netconf base-op-authorization**[10].

Protocol operation	Example	Supported	Notes
	]]>]]>		
	<target><**startup/**></target>	Yes	—
	<target><**url/**></target>	Yes	—
	<target><**running/**></target>	No	—
	<target><**candidate/**></target>	No	—
discard-changes	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id="101" xmlns:nc= "urn:ietf:params:xml:ns:netconf: base:1.0"> <discard-changes/> </rpc>]]>]]>	Yes	To use the discard-changes operation, the user must belong to a command-authorization profile with **discard-changes** enabled under **netconf base-op-authorization**[10].
	<discard-changes> **<configuration-region xmlns= "urn:nokia.com:sros:ns:yang:sr:ietf- netconf-augments">...</ configuration-region>** </discard-changes>	Yes	Optional. Specify "bof", "configure", "debug", or "li". The default, if not specified, is "configure".
create-subscription	<?xml version="1.0" encoding="UTF-8"?> <rpc message-id="101" xmlns:nc= "urn:ietf:params:xml:ns:netconf: base:1.0"> <create-subscription/> </rpc>]]>]]>	Yes	To use the create-subscription operation, the user must belong to a command-authorization profile with **create-subscription** enabled under **netconf base-op-authorization**[10]. The configuration of **base-op-authorization create-subscription** is not pre-emptive, which means it is checked only at the time of the initial subscription. Configuration changes to the **base-op-authorization** do not cancel any in-progress subscriptions and

Protocol operation	Example	Supported	Notes
			operators who successfully subscribed continue to receive messages.
	`<create-subscription><stream/></create-subscription>`	Yes	Optional
	`<create-subscription><startTime/></create-subscription>`	Yes	Optional
	`<create-subscription><stopTime/></create-subscription>`	Yes	Optional
	`<create-subscription><filter/></create-subscription>`	No	Optional

get-schema	`<?xml version="1.0" encoding="UTF-8"?>`
	`<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">`
	`<get-schema xmlns= "urn:ietf:params:xml:ns:yang:ietfnetconf-monitoring">`
	`</get-schema>`
	`</rpc>`
	`]]>]]>`
		Yes	To use the get-schema operation, the user must belong to a command-authorization profile with **get-schema** enabled under **netconf base-op-authorization**[10].

	`<get-schema><identifier/></get-schema>`	Yes	Mandatory
	`<get-schema><version/></get-schema>`	Yes	Optional
	`<get-schema><format/></get-schema>`	Yes	Optional

get-data	`<?xml version="1.0" encoding="UTF-8"?>`
	`<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">`
	`<get-data>`
	`</get-data>`
	`</rpc>`
		Yes	To use the get-data operation, the user must belong to a command-authorization profile with **get-data** enabled under **netconf base-op-authorization**[10].

Protocol operation	Example	Supported	Notes
	]]>]]>		
	<get-data><**datastore/**></get-data>	Yes	Mandatory The "startup" cannot be used.
	<get-data><**subtree-filter/**></get-data>	Yes	Optional
	<get-data><**max-depth/**></get-data>	Yes	Optional
	<get-data> <origin-filter xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"> </origin-filter> </get-data>	Yes	Optional Available on the "operational" datastore only. The origin should be referenced using the defined namespace (as it is an identityref), for example **or:default**. The **origin-filter** option may be provided more than once. The **origin-filter** and **negated-origin-filter** options are mutually exclusive.
	<get-data> <negated-origin-filter xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"> </negated-origin-filter> </get-data>	Yes	Optional Available on the "operational" datastore only. The origin should be referenced using the defined namespace (as it is an identityref), for example **or:default**. The **negated-origin-filter** option may be provided more than once. The **origin-filter** and **negated-origin-filter**

Protocol operation	Example	Supported	Notes
			options are mutually exclusive.
	`<get-data><origin-filters/></get-data>`	No	Optional
	`<get-data><with-origin/></get-data>`	Yes	Optional. To use this option the datastore must be “operational”.
	`<get-data><with-defaults/></get-data>`	Yes	Optional
	`<get-data><nmdaaug:with-managed/></get-data>`	Yes	Optional. To use this option the datastore must be "operational".

	`<get-data>`
	`<nmdaaug:managed-filter>`
	`</nmdaaug:managed-filter>`
	`</get-data>`
		Yes	Optional. To use this option the datastore must be "operational".

	`<get-data><xpath-filter/></get-data>`	No	Optional

	`<get-data>`
	`<configuration-region xmlns=`
	`"urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">`
	`</configuration-region>`
	`</get-data>`
		Yes	Optional.
			Specify "bof", "configure", "debug", or "li". The default, if not specified, is "configure".

action	`<?xml version="1.0" encoding="UTF-8"?>`
	`<rpc message-id="101" xmlns=`
	`"urn:ietf:params:xml:ns:netconf:base:1.0">`
	`<action xmlns=`
	`"urn:ietf:params:xml:ns:yang:1">`
	`<admin xmlns=`
	`"urn:nokia.com:sros:ns:yang:sr:oper-admin">`
	`<redundancy>`
	`<synchronize>`
	`<configuration/>`
	`</synchronize>`
	`</redundancy>`
	`</admin>`
	`</action>`
	`</rpc>`
		Yes	See [11]

[11] YANG 1.1 defines an <action> element. See Individually YANG-modeled operations for information about SR OS support for actions.

Protocol operation	Example	Supported	Notes
	]]>]]>		

5.2.3.1 <get>

A <get> request can retrieve both the configuration and state data.

If any nodes from the configure tree are included in a <get> request filter, at minimum, the <configure> tag must contain a namespace. If the namespace is not specified, SR OS returns an error.

SR OS allows the use of the empty namespace, shown as xmlns="", to mean any namespace.

A <get> request is analyzed for syntax errors before it is executed. If a syntax error is found, a single global <rpc-error> for the entire request is sent in the reply.

Responses are provided for each item in the request until the first item with an error is found. A <response> tag containing the error information, followed by an <rpc-error> tag (and sub-tags) is attached to the erroneous item. The reply is returned, and no subsequent items are executed.

For a non-syntax error, the <rpc-error> for an item is placed after the </response> information and not included in the <response> tag.

To retrieve BOF configuration, the "bof" <configuration-region> must be specified within the <get> RPC.

Example: BOF configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
    bof
    </configuration-region>
    <filter>
        <bof xmlns="urn:nokia.com:sros:ns:yang:sr:bof-conf">
        </bof>
    </filter>
</get>
</rpc>
]]>]]>
```

To retrieve the debug configuration, the "debug" <configuration-region> must be specified within the <get> RPC.

Example: debug configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
   <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
    debug
   </configuration-region>
   <filter>
      <debug xmlns="urn:nokia.com:sros:ns:yang:sr:debug-conf">
      </debug>
   </filter>
</get>
</rpc>
```

```
]]>]]>
```

To retrieve the LI configuration, the "li" <configuration-region> must be specified within the <get> RPC.

Example: LI configuration

```
<get>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      li
    </configuration-region>
    <filter>
        <li xmlns="urn:nokia.com:sros:ns:yang:sr:li-conf">
       </li>
    </filter>
</get>
```

When a <configuration-region> is not specified, the <configuration-region> (that is, the main non-LI configuration region) is considered to be "configure" by default.

When a mismatched namespace or <configuration-region> combination is specified, SR OS returns an empty <data>.

If present, configuration annotations are encoded with the nokia-attr:comment attribute.

Example: Annotated configuration for the system name

In the following example, **configure system name** is annotated with the comment "This is a comment on the system name." in the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
            <system>
                <name nokia-attr:comment="This is a comment on the system name."> node2</
name>
            </system>
        </configure>
    </data>
</rpc-reply>
```

The optional proprietary <format> argument is supported. See Output format selection for more information.

See the following sections for examples of <get> request and response messages:

- Namespace specified in <configure> tag
- Namespace error in <configure> tag
- Namespace specified in <state> tag
- Namespace error in <state> tag
- Output format tag

See the <get-config> section for details about subtree filtering support.

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

5.2.3.1.1 Namespace specified in <configure> tag

Example: <configure> tag containing a namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
    <filter>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
        </configure>
    </filter>
</get>
</rpc>
]]>]]>
```

Example: Reply which returns no errors

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
        </configure>
    </data>
</rpc-reply>
]]>]]>
```

5.2.3.1.2 Wildcard namespace in <configure> tag

Example: <configure> tag containing a wildcard namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="get" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
      <configure xmlns="">
      </configure>
    </filter>
  </get>
</rpc>
```

Example: Reply which returns no errors

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="get" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
        </configure>
    </data>
</rpc-reply>
```

5.2.3.1.3 Namespace error in <configure> tag

The following example shows a <configure> tag that does not contain a namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
    <filter>
        <configure>
            <python xmlns="urn:nokia.com:sros:ns:yang:sr:conf-python">
            </python>
        </configure>
    </filter>
</get>
</rpc>
]]>]]>
```

The following example shows the reply, which returns SR OS errors.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <rpc-error>
        <error-type>protocol</error-type>
        <error-tag>bad-element</error-tag>
        <error-severity>error</error-severity>
        <error-message>
            Element is not valid in the specified context.
        </error-message>
        <error-info>
            <bad-element>configure</bad-element>
        </error-info>
    </rpc-error>
</rpc-reply>
]]>]]>
```

5.2.3.1.4 Namespace specified in <state> tag

Example: <state> tag that contains a namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
    <filter>
        <state xmlns="urn:nokia.com:sros:ns:yang:sr:state">
        </state>
    </filter>
</get>
</rpc>
]]>]]>
```

Example: Reply which returns no errors

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <state xmlns="urn:nokia.com:sros:ns:yang:sr:state">
...
```

```
...
        </state>
    </data>
</rpc-reply>
]]>]]>
```

5.2.3.1.5 Namespace error in <state> tag

Example: <state> tag not containing a namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
    <filter>
        <state>
        </state>
    </filter>
</get>
</rpc>
]]>]]>
```

Example: Reply which returns SR OS errors

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <rpc-error>
        <error-type>protocol</error-type>
        <error-tag>bad-element</error-tag>
        <error-severity>error</error-severity>
        <error-message>
            Element is not valid in the specified context.
        </error-message>
        <error-info>
            <bad-element>state</bad-element>
        </error-info>
    </rpc-error>
</rpc-reply>
]]>]]>
```

5.2.3.1.6 Output format tag

Example: MD-CLI format request and reply

The following example shows the <get> operation combined with a subtree filter and requests the output in the MD-CLI format.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
  <get>
    <format xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">md-cli</format>
    <filter>
      <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
        <system>
          <management-interface>
            <netconf/>
          </management-interface>
        </system>
```

```
      </configure>
    </filter>
  </get>
</rpc>
]]>]]>
```

The following example shows the reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <data>
    configure {
        system {
            management-interface {
                netconf {
                    admin-state enable
                    auto-config-save true
                    capabilities {
                        candidate true
                    }
                }
            }
        }
    }
    </data>
</rpc-reply>
]]>]]>
```

Example: JSON IETF format request and reply

The following example shows the <get> operation combined with a subtree filter and requests the output in the JSON IETF format.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
  <get>
    <format xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">json</format>
    <filter>
      <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
        <system>
          <management-interface>
            <netconf/>
          </management-interface>
        </system>
      </configure>
    </filter>
  </get>
</rpc>
]]>]]>
```

The following example shows the reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <data>
{
    "nokia-conf:configure": {
        "system": {
            "management-interface": {
                "netconf": {
```

```
                    "admin-state": "enable",
                    "auto-config-save": true,
                    "capabilities": {
                        "candidate": true
                    }
                }
            }
        }
    }
}
    </data>
</rpc-reply>
]]>]]>
```

5.2.3.2 <get-config>

A <get-config> operation is supported on the <running> and <candidate> datastores.

The <get-config> requests on the <candidate> datastore return only XML-formatted content. If no filter is specified for the <candidate> datastore, the SR OS returns only the Nokia SR OS configurations.

For the Nokia SR OS YANG modules, <get-config> requests that specify a nonexistent list node or presence container result in an <rpc-error> response.

For the "report-all" value with the <with-defaults> tag (RFC 6243) in an XML-content layer, <get-config> returns the equivalent of the CLI **info detail** command. The returned data includes attributes that are set to their default values. See Reply with defaults for more information.

Subtree filtering is supported for XML content layer <get-config> and <get> requests. The following is the subtree filtering behavior:

- Containment nodes are supported (as per section 6.2.3 of RFC 6241). Nodes that contain children nodes (containers) can be used for subtree filtering. See Containment node for more information.
- Attribute match expressions, as defined in section 6.2.2 of RFC 6241, are not supported.
- Selection nodes are supported, as defined in section 6.2.4 of RFC 6241. Empty leaf nodes and list name nodes can be used as selection nodes. A selection node that is a list and does not have a key specified is supported. See List without a key specified for more information.
- Content match nodes are supported, as defined in section 6.2.5 of RFC 6241). Content match nodes that are leafs but not keys are also supported. See Non-key leaf specified as selection node for more information.
- Wildcard namespaces, according to RFC6241, are supported with subtree filters in the <get-config> operation.

The <get-config> operation, by default, retrieves the data nodes that were set by a client to their default values for the Nokia SR OS models (the "explicit" mode, as defined in RFC 6243).

To retrieve BOF configuration, the "bof" <configuration-region> must be specified within the <get-config> <source>.

Example: Retrieving BOF configurations

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
  <source>
     <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
       bof
```

```
     </configuration-region>
     <running/>
  </source>
    <filter>
        <bof xmlns="urn:nokia.com:sros:ns:yang:sr:bof-conf"/>
    </filter>
</get-config>
</rpc>
]]>]]>
```

To retrieve the debug configuration, the "debug" <configuration-region> must be specified within the <get-config> <source>. For example:

Example: Retrieving debug configurations

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
  <source>
     <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
       debug
     </configuration-region>
     <running/>
  </source>
  <filter>
      <debug xmlns="urn:nokia.com:sros:ns:yang:sr:debug-conf"/>
  </filter>
</get-config>
</rpc>
]]>]]>
```

To retrieve the LI configuration, the "li" <configuration-region> must be specified within the <get-config> <source>. For example:

```
<get-config>
  <source>
    <configuration-region xmlsns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      li
    </configuration-region>
    <candidate/>
  </source>
   <filter>
     <li xmlns="urn:nokia.com:sros:ns:yang:sr:li-conf">
      <log>
        <log-id>
          <name>33</name>
        </log-id>
      </log>
    </li>
    </filter>
</get-config>
```

Alternatively, the <source> can be specified in the format of "configuration-region"-"datastore". For example:

```
<get-config>
  <source>
    <li-candidate xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </source>
  <filter>
    <li xmlns="urn:nokia.com:sros:ns:yang:sr:li-conf">
```

```
      <log>
        <log-id>
          <name>33</name>
        </log-id>
      </log>
    </li>
  </filter>
</get-config>
```

When both the <configuration-region> and the "configuration-region"-"datastore" format are used, SR OS applies the last tag used in the XML request. For example:

```
<get-config>
  <source>
    <configuration-region xmlsns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      configure
    </configuration-region>
      <li-candidate xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </source>
    <filter>
      …
    </filter>
</get-config>
```

In the preceding example, the <get-config> is used to retrieve the "li" configuration data from the "li" candidate datastore.

When a mismatched namespace or <configuration-region> combination is specified, SR OS returns an empty <data>.

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

If present, configuration annotations are encoded with the nokia-attr:comment attribute.

Example: Configuration annotations

In the following example, `configure system name` is annotated with the comment "This is a comment on the system name." in the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
            <system>
                <name nokia-attr:comment="This is a comment on the system name."> node2</
name>
            </system>
        </configure>
    </data>
</rpc-reply>
```

The optional proprietary <format> argument is supported. See Output format selection for more information.

See the following sections for examples of <get-config> request and response messages:

- Reply with defaults
- Reply without default values

- Containment node
- List without a key specified
- Non-key leaf specified as selection node
- Non-key leaf specified as a content match node
- Content match node on a list key
- Content match node on a leaf-list
- Output format tag

5.2.3.2.1 Reply with defaults

The following example shows the use of <with-defaults> with a value of "report-all".

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
    <source>
        <candidate/>
    </source>
    <filter>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
            <system>
                <security>
                    <cpm-filter>
                        <ipv6-filter>
                        </ipv6-filter>
                    </cpm-filter>
                </security>
            </system>
        </configure>
    </filter>
    <with-defaults xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">
      report-all
    </with-defaults>
</get-config>
</rpc>
]]>]]>
```

The following example shows the reply, which returns all attributes, even those with default values.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
            <system>
                <security>
                    <cpm-filter>
                        <ipv6-filter>
                            <admin-state>disable</admin-state>
                        </ipv6-filter>
                    </cpm-filter>
                </security>
            </system>
        </configure>
    </data>
</rpc-reply>
```

```
]]>]]>
```

5.2.3.2.2 Reply without default values

The following example shows a <get-config> request using <with-defaults>.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
    <source>
        <candidate/>
    </source>
    <filter>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
            <system>
                <security>
                <cpm-filter>
                    <ipv6-filter>
                    </ipv6-filter>
                    </cpm-filter>
                </security>
            </system>
        </configure>
    </filter>
</get-config>
</rpc>
]]>]]>
```

The following output shows the reply, which does not return attributes with default values.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
    <source>
        <candidate/>
    </source>
    <filter>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
            <system>
                <security>
                <cpm-filter>
                    <ipv6-filter>
                    </ipv6-filter>
                    </cpm-filter>
                </security>
            </system>
        </configure>
    </filter>
</get-config>
</rpc>
]]>]]>
```

5.2.3.2.3 Containment node

Example: Containment node

```
    <?xml version="1.0" encoding="UTF-8"?>
```

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
    <source><running/></source>
         <filter>
             <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                 <router/>
             </configure>
         </filter>
</get-config>
</rpc>
]]>]]>
```

5.2.3.2.4 List without a key specified

Example: Selection node as list without a specified key

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source><running/></source>
        <filter>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                <router>
                    <interface>
                    </interface>
                </router>
            </configure>
        </filter>
    </get-config>
</rpc>
]]>]]>
```

5.2.3.2.5 Non-key leaf specified as selection node

Example: Selection node list with non-key leaf specified

The following example shows a list with a non-key leaf specified as a selection node. Keys should be returned as well.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source><running/></source>
        <filter>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                <router>
                    <interface>
                        <admin-state/>
                    </interface>
                </router>
            </configure>
        </filter>
    </get-config>
</rpc>
]]>]]>
```

5.2.3.2.6 Non-key leaf specified as a content match node

Example: Non-key leaf specified as a content match node output

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source><running/></source>
        <filter>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                <router>
                    <interface>
                        <admin-state>disable</admin-state>
                    </interface>
                </router>
            </configure>
        </filter>
    </get-config>
</rpc>
]]>]]>
```

5.2.3.2.7 Content match node on a list key

Multiple key leafs for the same key cannot be requested inside the same instance of the list name node. Each key value must be inside its own instance of the list name node; for example, `<interface><interface-name>abc</interface-name></interface><interface><interface-name>def</interface-name></interface>`.

Example: Content match node on a list key

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source><running/></source>
        <filter>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                <router>
                    <interface>
                        <interface-name>Test</interface-name>
                    </interface>
                </router>
            </configure>
        </filter>
    </get-config>
</rpc>
]]>]]>
```

5.2.3.2.8 Content match node on a leaf-list

A content match node can be performed on a leaf-list, but SR OS requires that all leaf-list elements and nodes must be specified.

The full configuration (equivalent to the classic CLI command **admin display-config** or the MD-CLI command **admin show configuration**) can be obtained using a <get-config> request both when a <filter> tag is not present and when the <configure> tag is present inside a <filter> tag.

Example

The following example shows a content match node on a lead-list when the <filter> tag is not present.

```
    <?xml version="1.0" encoding="UTF-8"?>
    <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get-config>
            <source><running/></source>
        </get-config>
    </rpc>
    ]]>]]>

    <?xml version="1.0" encoding="UTF-8"?>
    <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get-config>
            <source><candidate/></source>
        </get-config>
    </rpc>
    ]]>]]>
```

Example

The following example shows a content match node on a lead-list when only the <configure> tag is present inside the <filter> tag.

```
    <?xml version="1.0" encoding="UTF-8"?>
    <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get-config>
            <source><running/></source>
            <filter>
                <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf"/>
            </filter>
        </get-config>
    </rpc>
    ]]>]]>

    <?xml version="1.0" encoding="UTF-8"?>
    <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get-config>
            <source><candidate/></source>
            <filter>
                <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf"/>
            </filter>
        </get-config>
    </rpc>
    ]]>]]>
```

5.2.3.2.9 Wildcard namespace in <configure> tag

Example: <configure> tag containing a wildcard namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="get-config" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <configure xmlns="">
```

```
      </configure>
    </filter>
  </get-config>
</rpc>
```

Example: Reply which returns no errors

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="get-config" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
        </configure>
    </data>
</rpc-reply>
```

5.2.3.2.10 Output format tag

Example: MD-CLI format request and reply

The following example shows the <get-config> operation combined with a subtree filter and requests the output in the MD-CLI format.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
  <get-config>
    <format xmlsns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">md-cli</format>
    <source><running/></source>
    <filter>
      <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
        <system>
          <management-interface>
            <netconf/>
          </management-interface>
        </system>
      </configure>
    </filter>
  </get-config>
</rpc>
]]>]]>
```

The following example shows the reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <data>
    configure {
        system {
            management-interface {
                netconf {
                    admin-state enable
                    auto-config-save true
                    capabilities {
                        candidate true
                    }
                }
            }
        }
```

```
    }
    </data>
</rpc-reply>
]]>]]>
```

Example: JSON IETF format request and reply

The following example shows the <get-config> operation combined with a subtree filter and requests the output in the JSON IETF format.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
  <get-config>
    <format xmlsns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">json</format>
    <source><running/></source>
    <filter>
      <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
        <system>
          <management-interface>
            <netconf/>
          </management-interface>
        </system>
      </configure>
    </filter>
  </get-config>
</rpc>
]]>]]>
```

The following example shows the reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <data>
{
    "nokia-conf:configure": {
        "system": {
            "management-interface": {
                "netconf": {
                    "admin-state": "enable",
                    "auto-config-save": true,
                    "capabilities": {
                        "candidate": true
                    }
                }
            }
        }
    }
}
    </data>
</rpc-reply>
```

5.2.3.3 <edit-config>

An <edit-config> operation is supported on the <candidate> datastore. If <edit-config> requests specify the <running> datastore as a target with the Nokia SR OS YANG modules "urn:nokia.com:sros:ns:yang:sr:conf" namespace, it results in an error response.

The <edit-config> requests to the <candidate> datastore only result in XML- formatted content.

An internal "implicit" lock is in place on the <running> datastore that includes all configuration commands in SR OS, and not just the "urn:nokia.com:sros:ns:yang:sr:conf" namespace. The following actions affect the "implicit" lock:

- The first NETCONF <edit-config> on a global <candidate> datastore triggers the "implicit" lock.
- The completion of a NETCONF <commit> releases the "implicit" lock.
- The NETCONF <discard-changes> operation releases the "implicit" lock.

The following scenarios are impacted when the "implicit" lock is in place:

- In model-driven configuration mode, classic CLI commands are blocked and SR OS returns an error.
- In model-driven configuration mode, an SNMP set request is blocked and SR OS returns an error.

One or more <edit-config> requests can be performed on the <candidate> datastore before changes are committed or discarded.

The supported <edit-config> operation attribute values are listed in the following table.

Table 26: <edit-config> operation attribute values

Command	Notes
urn:nokia.com:sros:ns:yang:sr:conf namespace Nokia SR OSYANG modules	
merge (Nokia SR OS modules)	Supported
remove (Nokia SR OS modules)	A remove operation removes the deleted configuration and returns it to the default value. A remove operation automatically removes all child objects of a deleted object (leaves, lists, containers, and so on). Explicit shutdown of the object being removed (or any child) is not required and results in an error if a merge operation is specified on a tag that inherits a remove operation. A remove operation is allowed on non-presence containers. The non-presence container and all of its children are removed (for example, a non-presence container with no child nodes is not displayed in a <get> or <get-config> reply). A remove operation is allowed on an object where all child branches and dependencies are automatically removed (but the remove operation fails if any outside objects refer to the object being removed). A remove operation is allowed on a <shutdown/> leaf (which returns it to its default value). A remove operation is allowed on a non-Boolean leaf. Upon specifying a remove operation on a node where none of its children belong to the urn:nokia.com:sros:ns:yang:sr:conf namespace (the Nokia SR OS YANG modules), SR OS does not return an error and completes the node removal.

Command	Notes
	A remove operation for a leaf, where the request also specifies a value for the leaf, results in an error.
delete (Nokia SR OS modules)	SR OS returns an error if a delete operation is performed on a list that does not specify a key (that is, an attempt to delete all members of a list). SR OS returns an error if a delete operation is performed on a leaf or presence container that is already deleted (or has the default value and the default-handling is trim). SR OS may return an error and may not complete the deletion operation when a delete operation is performed on a node where any of its children do not belong to the urn:nokia.com:sros:ns:yang:sr:conf namespace (the Nokia SR OS YANG modules). A delete operation removes the deleted configuration and returns it to the default value. A delete operation automatically deletes all child objects of a deleted object (leaves, lists, containers, and so on). Explicit shutdown of the object being deleted (or any of its children) is not required and results in an error if a merge operation is specified on a tag that inherits a delete operation. A delete operation is allowed on non-presence containers. The non-presence container and all of its children are deleted (for example, a non-presence container with no child nodes is not displayed in a <get> or <get-config> reply). A delete operation is allowed on an object where all child branches and dependencies are automatically deleted (but the delete operation fails if any outside objects refer to the object being deleted). A delete operation is allowed on a <shutdown/> leaf (which returns it to its default value). A delete operation is allowed on a non-Boolean leaf. Upon specifying a delete operation on a node where none of its children belong to the urn:nokia.com:sros:ns:yang:sr:conf namespace (the Nokia SR OS YANG modules), SR OS does not return an error and completes the node deletion. A delete operation for a leaf, where the request also specifies a value for the leaf, results in an error.
create (Nokia SR OS modules)	When a create operation for a leaf or presence container is performed, SR OS returns an error if the leaf or presence container is being set to the same value (unless the default-handling is trim and the value being set is the default value).
replace (Nokia SR OS modules)	Supported

The <edit-config> operation <default-operation> parameter is supported with the following values:

- replace
- merge
- none

  In the Nokia SR OS YANG modules "urn:nokia.com:sros:ns:yang:sr:conf namespace", an operation of "none" (inherited or direct) on a leaf node that does not exist in the data model causes SR OS to return an error with an <error-tag> value of data-missing.

For delete and remove operations in the Nokia SR OS namespace, the SR OS NETCONF server will recursively unwind any children of the node being deleted or removed first before removing the node. The deepest child branch of the request is examined first, and any leaves are processed, after which the server works backwards out of the deepest branches back up to the object where the delete operation was specified.

The following applies to the Nokia SR OS YANG modules "urn:nokia.com:sros:ns:yang:sr:conf" namespace.

- SR OS returns an error if an explicitly defined <edit-config> operation (such as "delete") is specified on a "key" leaf.
- The "operation" attribute is inherited from the parent node if not explicitly specified (same as namespaces). If no parent node is available, the "default-operation" value is used, meaning that the "operation" attribute has a "scope" that it applies to the nested nodes until it is redefined.

See Application of default operation value for parent and child nodes and Exceptions to the default operation handling for more information.

The following scenarios simplify "operation" inheritance, where the first line in each scenario represents the operation value of the parent node and the following lines represent the possible operation values for the child nodes and the SR OS behavior in each case:

- **Create**

  Create/Merge: SR OS processes the request, which succeeds or fails based on the behavior of this operation.

  Delete/Remove: SR OS returns an error.

- **Merge**

  Create/Merge/Delete/Remove: SR OS processes the request, which succeeds or fails based on the behavior of this operation.

- **Delete/Remove**

  Create/Merge: SR OS returns an error.

  Delete/Remove: SR OS processes the request, which succeeds or fails based on the behavior of this operation.

The <error-option> is supported. SR OS implements the rollback-on-error behavior at all times, when:

- the error-option is not specified
- the error-option is specified and set to either stop-on-error or rollback-on-error

As per RFC 6020, *YANG- A Data Modeling Language for the Network Configuration Protocol (NETCONF)*, the "insert" and "value" attributes are supported with user-ordered leaf-lists to insert or move a user-ordered leaf-list entry in the candidate datastore.

As per RFC 6020, the "insert" and "key" attributes are supported with user-ordered lists to insert or move a user-ordered list entry in the candidate datastore.

With a NETCONF <edit-config> RPC, SR OS authorizes all configuration changes in the <candidate> datastore; that is, it checks the YANG tree and authorizes every changed Managed Object (MO).

The deletion of a container results in the deletion of any children containers that are authorized for deletion, as well as their contents. Children containers that are not authorized for deletion, as well as their contents, are retained. For example, upon deletion of **configure system**, **configure system security** is not deleted because the deletion of that child container is not authorized.

Note: A "no change" for a value does not require authorization. Therefore, it is possible to execute a non-authorized command if there is no change in value.
For example, when a user is not authorized to change **access li**, but attempts to change it for another a user who already has **access li**, SR OS allows that action because there is no change in value.

To edit BOF configuration, the "bof" <configuration-region> must be specified within the <edit-config> <target>.

Example: Edited BOF configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
  <target>
       <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
         bof
       </configuration-region>
    <candidate/>
  </target>
    <config>
          <bof xmlns="urn:nokia.com:sros:ns:yang:sr:bof-conf">
               <console>
                     <wait-time>5</wait-time>
               </console>
          </bof>
    </config>
</edit-config>
</rpc>
]]>]]>
```

To edit the debug configuration, the "debug" <configuration-region> must be specified within the <edit-config> <target>. For example:

Example: Edited debug configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
  <target>
       <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-
augments">debug</configuration-region>
    <candidate/>
  </target>
    <config>
      <!-- place debug configuration changes here -->
    </config>
</edit-config>
</rpc>
]]>]]>
```

To edit the LI configuration, the "li" <configuration-region> must be specified within the <edit-config> <target>. For example:

Example: Edited LI configuration

```
<edit-config>
  <target>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      li
    </configuration-region>
    <candidate/>
  </target>
  <config>
    <!-- place LI configuration changes here -->
  </config>
</edit-config>
```

Alternatively, the <target> can be specified in the format of "configuration-region"-"datastore". For example:

```
<edit-config>
  <target>
    <li-candidate xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </target>
  <config>
    <!-- place LI configuration changes here -->
  </config>
</edit-config>
```

When both the <configuration-region> and the "configuration-region"-"datastore" format are used, SR OS applies the last tag used in the XML request. For example:

```
<edit-config>
<target>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      configure
    </configuration-region>
    <li-candidate xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
</target>
<config>
    <!-- place LI configuration changes here -->
</config>
</edit-config>
```

When a mismatched namespace or <configuration-region> combination is specified, SR OS returns an error.

The <edit-config> RPC can only be used to push LI configuration changes if all of the following conditions are true.

- The NETCONF user is an LI user.
- The NETCONF session has an exclusive lock on the LI configuration region and <candidate> datastore.
- The specified <configuration-region> is "li".
- The YANG modules that are used are the Nokia SR OS YANG modules.

If any of the preceding conditions is false, SR OS returns an error.

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

Configuration annotations may be encoded with the nokia-attr:comment attribute to add, change, or delete the annotations. In the following example, **configure system name** is annotated with the comment "This is a comment on the system name." in the <edit-config> RPC:

Example: Annotated configuration output

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target><candidate/></target>
    <config>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
            <system>
                <name nokia-attr:comment="This is a comment on the system name."> node2</
name>
            </system>
        </configure>
    </config>
  </edit-config>
</rpc>
]]>]]>
```

See the following sections for examples of <edit-config> request and response messages:

- <running> datastore with the "urn:nokia.com:sros:ns:yang:sr:conf" namespace
- Application of default operation value for parent and child nodes
- Exceptions to the default operation handling

5.2.3.3.1 <running> datastore with the "urn:nokia.com:sros:ns:yang:sr:conf" namespace

The following example shows the use of the <running> datastore with the "urn:nokia.com/sros:ns:yang:sr:conf" namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
        <target><running/></target>
        <config>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                <python>
                    <python-script>
                        <script-name>testing</script-name>
                    </python-script>
                </python>
            </configure>
        </config>
    </edit-config>
</rpc>
]]>]]>
```

The following example shows the reply, which returns SR OS errors.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <rpc-error>
        <error-type>protocol</error-type>
        <error-tag>operation-not-supported</error-tag>
        <error-severity>error</error-severity>
```

```
        <error-message>
            writable-running capability is not supported
        </error-message>
        <error-info>
            <bad-element>running</bad-element>
        </error-info>
    </rpc-error>
</rpc-reply>
]]>]]>
```

5.2.3.3.2 Application of default operation value for parent and child nodes

The following example shows that the default (operation=”merge”) applies to all parent and child nodes.

Example: Default merge

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
    <target><candidate/></target>
    <config>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
            <service>
                <epipe>
                    <service-name>CustDoc</service-name>
                    <customer>1</customer>
                    <description>Local epipe</description>
                </epipe>
         </service>
        </configure>
    </config>
</edit-config>
</rpc>
]]>]]>
```

Example: Output of the reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
]]>]]>
```

5.2.3.3.3 Exceptions to the default operation handling

The following examples show that the default (operation=”merge”) applies to all parent and child nodes except for <description>, which has a (operation="remove").

Example: Output of the default operation handling exception

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
    <target><candidate/></target>
    <config>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
```

```
            <service>
                <epipe>
                     <service-name>CustDoc</service-name>
                     <customer>1</customer>
                     <description nc:operation="remove">Local epipe</description>
                </epipe>
            </service>
        </configure>
    </config>
</edit-config>
</rpc>
]]>]]>
```

Example: Output of the reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc=
"urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
]]>]]>
```

5.2.3.4 <copy-config>

To <copy-config> BOF configuration, the "bof" <configuration-region> must be specified within the <copy-config> RPC <source> and <target>.

Example: Configuration usage

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
<copy-config>
    <target>
        <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-
augments">bof</configuration-region>
        <startup/>
    </target>
    <source>
        <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-
augments">bof</configuration-region>
        <running/>
    </source>
</copy-config>
</rpc>
]]>]]>
```

Note: When <copy-config> RPC is used with the "bof" <configuration-region>, only the following combinations are supported:

- <source>config</source> and <target>candidate</target>
- <source>running</source> and <target>startup</target>

To <copy-config> debug configuration, the "debug" <configuration-region> must be specified within the <copy-config> RPC <source> and <target>. For example:

Example: Configuration debug usage

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
<copy-config>
          <target>
                  <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-
augments">debug</configuration-region>
                  <candidate/>
          </target>
          <source>
                  <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-
augments">debug</configuration-region>
                  <config></config>
          </source>
</copy-config>
</rpc>
]]>]]>
```

Note: When <copy-config> RPC is used with the "debug" <configuration-region>, only the following combination is supported:

- <source><config></config></source> and <target><candidate/></target>

To <copy-config> LI configuration, the "li" <configuration-region> must be specified within the <copy-config> RPC <source> **and** <target>. When the <configuration-region> is "li" or "bof", SR OS can only <copy-config> from the <running> datastore to the <startup> datastore. For example:

Example: Configuration LI usage

```
<copy-config>
  <target>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      li
    </configuration-region>
    <startup/>
  </target>
  <source>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">li</
configuration-region>
    <running/>
  </source>
</copy-config>
```

Performing a <copy-config> between datastores from different configuration regions is not allowed. Mismatching the source or target <configuration-region> causes SR OS to return an error.

Alternatively, the <target>/<source> can be specified in the format of "configuration-region"-"datastore". For example:

```
<copy-config>
  <target>
    <li-startup xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </target>
  <source>
    <li-running xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </source>
</copy-config>
```

When both the <configuration-region> and the "configuration-region"-"datastore" format are used, SR OS applies the last tag used in the XML request. For example:

```
<copy-config>
  <target>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      configure
    </configuration-region>
    <li-startup xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </target>
  <source>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      configure
    </configuration-region>
    <li-running xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </source>
</copy-config>
```

In the preceding example, the <copy-config> is used to copy the configuration data from the "li" <running> datastore to the "li" <startup> datastore.

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

5.2.3.5 <delete-config>

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

5.2.3.6 <lock>

Taking the <candidate> datastore lock is equivalent to starting a CLI exclusive session. A NETCONF session cannot take the <candidate> datastore lock if there are uncommitted configuration changes in the <candidate> datastore.

It is recommended that a NETCONF session should always take the <candidate> datastore lock before reading or writing configurations to ensure the <candidate> datastore is not changed by other model-driven sessions. Release the <candidate> datastore lock after all configurations are successfully read or committed.

When either the <running> datastore lock or the <candidate> datastore lock is taken by a NETCONF session, the following applies:

- no NETCONF session can take the <running> datastore lock
- no NETCONF session can take the <candidate> datastore lock
- no other NETCONF session can perform an <edit-config> on the <running> datastore
- no other NETCONF session can perform an <edit-config> on the <candidate> datastore
- no other NETCONF session can perform a <commit> on the <candidate> datastore
- no other NETCONF session can perform a <discard-changes> on the <candidate> datastore
- CLI becomes read-only
- the classic CLI **rollback revert** command is blocked

A datastore lock is unlocked when a user disconnects a NETCONF session by using the **admin disconnect** command or **Ctrl-c**, or by performing a <kill-session> / <close-session> operation. Upon disconnecting a NETCONF session that had acquired a datastore lock, SR OS:

- releases the lock
- discards any "uncommitted" changes

Note: The behavior is different if the disconnected NETCONF session was using the global <candidate> datastore and had uncommitted configuration changes. In that case, SR OS keeps the "uncommitted" changes in the global <candidate> datastore.

Timeouts for locks are not supported. No specific **admin** or **tools** commands are provided to release the lock without disconnecting the session that holds it, but the session that holds the lock can be administratively disconnected through a CLI command to release the lock.

Using a CLI **show** command, the operator can determine whether the <running> datastore is locked, the <candidate> datastore is locked, or both are locked, and the session ID of the session that holds the lock.

From CLI, the operator can configure whether users that belong to a specific profile have permission to lock NETCONF sessions.

An active NETCONF session can be disconnected from the CLI using the session ID. The user can use the **show** command to find the NETCONF session ID, then use this session ID in the **admin** command to disconnect the NETCONF session.

To lock a BOF datastore, the "bof" <configuration-region> must be specified within the <lock> <target>.

Example: Lock a BOF datastore configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
          <lock>
                 <target>
                       <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-
netconf-augments">
                         bof
                       </configuration-region>
                       <candidate/>
                 </target>
          </lock>
</rpc>
]]>]]>
```

To lock a debug datastore, the "debug" <configuration-region> must be specified within the <lock> <target>. For example:

Example: Lock a debug datastore configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
          <lock>
                 <target>
                       <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-
netconf-augments">
                         debug
                       </configuration-region>
                       <candidate/>
                 </target>
          </lock>
```

```
</rpc>
]]>]]>
```

To lock an LI datastore, the "li" <configuration-region> must be specified within the <lock> <target>. For example:

Example: Lock an LI datastore configuration

```
<lock>
  <target>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      li
    </configuration-region>
    <candidate/>
  </target>
</lock>
```

Alternatively, the <target> can be specified in the format of "configuration-region"-"datastore". For example:

```
<lock>
  <target>
    <li-candidate xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </target>
</lock>
```

When both the <configuration-region> and the "configuration-region"-"datastore" format are used, SR OS applies the last tag used in the XML request. For example:

```
<lock>
  <target>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      configure
    </configuration-region>
    <li-candidate xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </target>
</lock>
```

In the preceding example, the <lock> is used to lock the "li" <candidate> datastore.

The LI datastores have independent locks from the main configuration datastores.

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

5.2.3.7 <unlock>

Because there is a single lock per datastore regardless of the scope of that lock, the following apply to the <unlock> operation:

- The <unlock> operation must be performed only on the <running> datastore to unlock it. An error results and the lock is not released if a different datastore is specified with the <unlock> operation.
- The <unlock> operation must be performed only on the <candidate> datastore to unlock it. An error results and the lock is not released if a different datastore is specified with the <unlock> operation.

Performing an <unlock> operation on the <candidate> datastore discards all uncommitted <candidate> datastore changes.

To unlock a BOF datastore, the "bof" <configuration-region> must be specified within the <unlock> <target>.

Example: Unlock a BOF datastore configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <unlock>
               <target>
                     <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-
netconf-augments">
                       bof
                     </configuration-region>
                     <candidate/>
               </target>
          </unlock>
</rpc>
]]>]]>
```

To unlock a debug datastore, the "debug" <configuration-region> must be specified within the <unlock> <target>. For example:

Example: Unlock a debug datastore configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <unlock>
               <target>
                     <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-
netconf-augments">
                       debug
                     </configuration-region>
                     <candidate/>
               </target>
          </unlock>
</rpc>
]]>]]>
```

To unlock an LI datastore, the "li" <configuration-region> must be specified within the <unlock> <target>. For example:

Example: Unlock an LI datastore configuration

```
<unlock>
  <target>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      li
    </configuration-region>
    <candidate/>
  </target>
</unlock>
```

Alternatively, the <target> can be specified in the format of "configuration-region"-"datastore". For example:

```
<unlock>
  <target>
    <li-candidate xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </target>
</unlock>
```

When both the <configuration-region> and the "configuration-region"-"datastore" format are used, SR OS applies the last tag used in the XML request. For example:

```
<unlock>
  <target>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      configure
    </configuration-region>
    <li-candidate xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </target>
</unlock>
```

In the preceding example, the <unlock> is used to unlock the "li" <candidate> datastore.

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

5.2.3.8 <commit>

The following are the characteristics of the <commit> operation:

- It is the equivalent of the **commit** CLI command.
- If a <commit> operation fails and more than one error exists, SR OS returns multiple errors.
- If SR OS is unable to commit all changes in the <candidate> datastore, the <running> datastore is left unchanged.
- When a NETCONF session is disconnected in the middle of a <commit> operation (using the CLI command, Ctrl-c, or <kill-session>), SR OS keeps the <running> datastore unchanged.
- The persistence of changes made using a <commit> operation is operator-controlled. A copy of the <running> datastore to the <startup> datastore can be automatically performed after each successful <commit> operation. This behavior can be enabled or disabled through a CLI command.
- Changes that exist in the <candidate> datastore before they are committed to the <running> datastore impact the following scenarios:
  - a CLI user trying to make immediate changes, because SR OS may block immediate CLI configurations
  - an SNMP set request, because SR OS may block the request and return an error

To commit BOF configuration, the "bof" <configuration-region> must be specified within the <commit> RPC.

Example: Commit BOF configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<commit>
      <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
        bof
      </configuration-region>
</commit>
</rpc>
]]>]]>
```

To commit the debug configuration, the "debug" <configuration-region> must be specified within the <commit> RPC. For example:

Example: Commit debug configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<commit>
      <configuration-region
xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      debug
      </configuration-region>
</commit>
</rpc>
]]>]]>
```

To commit the LI configuration, the "li" <configuration-region> must be specified within the <commit> RPC. For example:

Example: Commit LI configuration

```
<commit>
  <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
    li
  </configuration-region>
</commit>
```

The <commit> RPC can only be used with LI configuration changes if all of the following conditions are true.

- The NETCONF user is a LI user.
- The NETCONF session has an exclusive lock on the LI configuration region and <candidate> datastore.
- The specified <configuration-region> is "li".
- The YANG modules used are the Nokia SR OS YANG modules.

If any of the preceding conditions are false, SR OS returns an error.

The :confirmed-commit capability cannot be used with LI configuration changes.

The :confirmed-commit capability is advertised in the SR OS NETCONF server <hello> as follows:

```
<capability>urn:ietf:params:netconf:capability:confirmed-commit:1.1</capability>
```

The :confirmed-commit capability includes the following characteristics.

- The capability is not advertised if the operator disables the <candidate> datastore capability using the available SR OS CLI command.
- The following table lists the supported parameters for a <commit> operation.

Table 27: Parameters for a <commit> operation

Parameter	Description
<comment>	Optional. Specifies a comment in the commit history.
<confirmed>	Indicates a confirmed <commit> operation.

Parameter	Description
<confirm-timeout>	Specifies the timeout period for confirmed commit (in seconds). If unspecified, the confirmed commit timeout defaults to 600 seconds (10 minutes).
<persist>	Configures the confirmed commit changes to survive a session termination. It sets a token on the ongoing confirmed commit. If <persist> is not in the confirmed commit operation, any follow-up commit and the confirming commit must be issued on the same session that issued the confirmed commit. If <persist> is in the confirmed commit operation, a follow-up commit and the confirming commit can be on any session. However, they must include a <persist-id> element with a value equal to the value of the <persist> element in the confirmed commit. The <persist> element cannot be changed through a follow-up confirmed commit.
<persist-id>	Issues a follow-up confirmed commit or the confirming commit from any session, using the same token from the <persist> element of the confirmed commit. The <persist-id> element cannot be changed through a follow-up confirmed commit.
<configuration-region>	Optional. Specifies the configuration-region to commit.

- If <persist> is specified in the :confirmed commit, the configuration changes are rolled back only if the timeout expires before a confirming commit is received. The confirming commit must include a <persist-id> tag with a value equal to the value of the <persist> tag contained in the confirmed commit.
- If the NETCONF session that has initiated the confirmed commit closes while waiting for the confirming commit (for example, disconnected), SR OS restores the configuration to its former state before the confirmed commit was issued. This is valid only if <persist> is not defined in the confirmed commit. If a follow-up confirmed commit is issued before the timer expires, the timer is reset to the new value.
- The confirming commit and the follow-up confirmed commit cannot introduce additional configuration changes.
- The <cancel-commit> operation is supported. It can cancel an ongoing confirmed commit (that is, cancel the timer and rollback the changes introduced in the confirmed commit).
- If the <persist> parameter is not specified, the <cancel-commit> operation must be issued from the same session that issued the confirmed commit.
- A confirmed commit should not be used to commit configuration mode change to the classic mode because SR OS will switch to the classic mode before the second commit is sent.

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

5.2.3.9 <discard-changes>

The <discard-changes> operation causes the <candidate> datastore to revert to match the <running> datastore and discard any uncommitted configuration changes.

To discard BOF configuration changes, the "bof" <configuration-region> must be specified within the <discard-changes> RPC.

Example: Discard BOF configuration changes

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <discard-changes>
        <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
          bof
        </configuration-region>
    </discard-changes>
</rpc>
]]>]]>
```

To discard debug configuration changes, the "debug" <configuration-region> must be specified within the <discard-changes> RPC. For example:

Example: Discard debug configuration changes

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <discard-changes>
                        <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-
netconf-augments">
                          debug
                        </configuration-region>
              </discard-changes>
</rpc>
]]>]]>
```

To discard LI configuration changes, the "li" <configuration-region> must be specified within the <discard-changes> RPC. For example:

Example: Discard LI configuration changes

```
<discard-changes>
  <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
    li
  </configuration-region>
</discard-changes>
```

The <discard-changes> RPC can only be used with LI configuration changes if all of the following conditions are true.

- The NETCONF user is a LI user.
- The NETCONF session has an exclusive lock on the LI configuration region and <candidate> datastore.
- The specified <configuration-region> is "li".
- The YANG modules used are the Nokia SR OS YANG modules.

If any of the preceding conditions are false, SR OS returns an error.

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

5.2.3.10 <validate>

The following support is provided for the :validate capability.

The validate:1.1 and :validate:1.0 capabilities are advertised in the NETCONF server <hello> as the follows.

- <capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
- <capability>urn:ietf:params:netconf:capability:validate:1.1</capability>

To validate BOF configuration, the "bof" <configuration-region> must be specified within the <validate> <source>.

Example: Validate BOF configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <validate>
      <source>
         <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-
augments">
           bof
         </configuration-region>
         <candidate/>
      </source>
   </validate>
</rpc>
]]>]]>
```

To validate the debug configuration, the "debug" <configuration-region> must be specified within the <validate> <source>. For example:

Example: Validate debug configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <validate>
              <source>
                        <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-
netconf-augments">
                          debug
                        </configuration-region>
                        <candidate/>
              </source>
   </validate>
</rpc>
]]>]]>
```

To validate the LI configuration, the "li" <configuration-region> must be specified within the <validate> <source>. For example:

Example: Validate LI configuration

```
<validate>
  <source>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
```

```
      li
    </configuration-region>
    <candidate/>
  </source>
</validate>
```

Alternatively, the <source> can be specified in the format of “configuration-region”-”datastore”. For example:

```
<validate>
  <source>
    <li-candidate xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </source>
</validate>
```

When both the <configuration-region> and the "configuration-region”-"datastore” format are used, SR OS applies the last tag used in the XML request. For example:

```
<validate>
  <source>
    <configuration-region xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments">
      configure
    </configuration-region>
    <li-candidate xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-augments"/>
  </source>
</validate>
```

In the preceding example, the <validate> is used on the "li” <candidate> datastore.

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

5.2.3.11 <get-schema>

A <get-schema> operation is supported for explicit schema retrieval via NETCONF. See NETCONF monitoring for more information.

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

5.2.3.12 <get-data>

A `<get-data>` operation is similar to a `<get-config>` operation. When applied to the `running` or `candidate` datastores, the `<get-data>` operation returns `config false` YANG nodes because the `running` and `candidate` datastores are configuration datastores (meaning they contain configuration data only and not state data).

You can apply a `<get-data>` operation to the `intended` datastore, but you cannot apply other non-NMDA compliant operations such as `<get>` and `<get-config>`. The `intended` datastore returns the data expanded with any templated configuration from configuration groups. The `intended` datastore is also a configuration datastore and returns `config false` YANG nodes only.

You can apply a `<get-data>` operation to the `operational` datastore, but you cannot apply other non-NMDA compliant operations, such as `<get>` and `<get-config>`. See Operational datastore for more information about the operational datastore.

The <get-data> operation supports the optional <config-filter> argument to restrict the output to config true or config false YANG nodes.

See Table 25: Protocol operations and level of support in Nokia SR OS NETCONF servers for more information.

The optional proprietary <format> argument is supported. See Output format selection for more information.

See Output format tag for an example of <get-data> request and response messages.

Wildcard namespaces, according to RFC6241, are supported with subtree filters in the <get-data> operation.

5.2.3.12.1 Output format tag

The following example shows the <get-data> operation combined with a subtree filter and requests the output in the MD-CLI format.

Example: MD-CLI format

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
  <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
            xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
    <format xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-nmda-augments">
      md-cli
    </format>
    <datastore>ds:running</datastore>
    <subtree-filter>
      <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
        <system>
          <management-interface>
            <netconf/>
          </management-interface>
        </system>
      </configure>
    </subtree-filter>
  </get-data>
</rpc>
]]>]]>
```

The following example shows the reply.

Example: MD-CLI format reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
    configure {
        system {
            management-interface {
                netconf {
                    admin-state enable
                    auto-config-save true
                    capabilities {
                        candidate true
                    }
                }
```

```
            }
        }
    }
    </data>
</rpc-reply>
]]>]]>
```

The following example shows the <get-data> operation combined with a subtree filter and requests the output in the JSON IETF format.

Example: JSON IETF format

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
  <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
            xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
    <format xmlns="urn:nokia.com:sros:ns:yang:sr:ietf-netconf-nmda-augments">
      json
    </format>
    <datastore>ds:running</datastore>
    <subtree-filter>
      <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
        <system>
          <management-interface>
            <netconf/>
          </management-interface>
        </system>
      </configure>
    </subtree-filter>
  </get-data>
</rpc>
]]>]]>
```

The following example shows the reply.

Example: JSON IETF format reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
{
    "nokia-conf:configure": {
        "system": {
            "management-interface": {
                "netconf": {
                    "admin-state": "enable",
                    "auto-config-save": true,
                    "capabilities": {
                        "candidate": true
                    }
                }
            }
        }
    }
}
    </data>
</rpc-reply>
]]>]]>
```

5.2.4 Datastore and operation combinations

The following table lists the operations supported by the datastores.

Table 28: Datastore and operation combinations

Operation	Datastore			
	<running>	<candidate>	<intended>	<operational>
<edit-config>		✓		
<get-config>	✓	✓		
<get-data>[12]	✓	✓	✓	✓

The <get> RPC returns data that has the config field set to false in the YANG model, and data from the <running> datastore that has the config field set to true in the YANG model.

5.2.5 Output format selection

The SR OS NETCONF implementation provides an optional proprietary <format> argument for the <get>, <get-config>, and <get-data> NETCONF operations and the <md-compare> YANG-modeled operation.

The <format> argument allows the operator to display the resultant data in either the XML (default), JSON IETF, or MD-CLI format.

The supported options for the <format> argument are as follows:

- xml
- json
- md-cli

Note: See <get>, <get-config>, and <get-data> for usage examples of the <format> option.

The supported options for the <format> argument are as follows for the <md-compare> YANG-modeled operation:

- xml
- md-cli
- md-cli-full

The md-cli option provides the output in a format similar to the MD-CLI command compare summary. The md-cli-full option provides the output in a format similar to the MD-CLI command compare.

Note: See <md-compare> YANG-modeled operation for usage examples.

12 The use of <get-data> and the <intended> or <operational> datastores require **nmda-support** to be enabled.

5.2.6 Private candidates over NETCONF

The SR OS NETCONF implementation provides a proprietary function to operate a specific NETCONF session in private candidate mode. This provides the ability for NETCONF (and CLI) sessions to cohabit with minimal disruption to each other because the sessions often configure entirely separate sections of the overall device configuration.

Setting up, configuring, and discarding private candidates

Sending the `urn:nokia.com:nc:pc` client capability from the NETCONF client to the NETCONF server in the <hello> message establishes the private candidate mode. The private candidate mode applies for the duration of a specific NETCONF session. The following example shows how to send the client capability.

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
    <capability>urn:nokia.com:nc:pc</capability>
  </capabilities>
</hello>
]]>]]>
```

The server sends no specific response to the <hello> message. From this point on, the session is in private candidate mode.

Private candidate mode operates in the same way as the default global candidate mode in many respects. Most of the RPCs and NETCONF operations that can be used remain the same.

Whenever a NETCONF operation references the candidate configuration datastore, a private candidate configuration is used. The system creates a private candidate configuration as a copy of the running configuration when the first RPC that requires a candidate configuration is used. The system also creates a baseline candidate configuration at this point. The baseline candidate configuration acts as a reference for identifying changes that the NETCONF client has made in the private candidate.

The following RPCs do not require a candidate and therefore do not trigger the creation of a candidate:

- <get-schema>
- <validate> when used without specifying a source configuration datastore
- any <action> RPC that does not require a candidate configuration

Every NETCONF session running in private candidate mode has its own private candidate configuration. If multiple NETCONF clients are used to configure specific sections of the configuration, the private candidate mode provides improvements in performance and operation because of the ability to edit configuration changes without locking the candidate configuration datastore.

See "Candidate configuration modes" in the *7705 SAR Gen 2 MD-CLI User Guide* for more information.

When a configuration is edited over NETCONF in the private candidate, MD-CLI users do not see these configuration changes in the global candidate configuration datastore as these changes are in the private candidate belonging to the specific NETCONF session.

When a NETCONF session running in private candidate mode sends a <commit> RPC, the SR OS node performs an update and subsequently a commit operation. See "Multiple simultaneous candidate configurations" in the *7705 SAR Gen 2 MD-CLI User Guide* for more information.

In the unlikely event that a collision in the configuration occurs because other users configure the same area of the configuration at the same time, the update operation in this process fails and therefore the <commit> RPC also fails. This update failure ensures the following:

- The user is aware of the conflict.
- The integrity of the operational configuration on the node remains intact.
- No unintended configuration changes resulting from any other users' input are made.

If the <commit> RPC fails for the above reason, sending the <commit> again may resolve the issue. If the issue is not resolved, the session must be restarted.

The <commit> RPC over a NETCONF session running in private candidate mode does not accept the **persist-id** option.

Closing a private candidate NETCONF session, either through disconnection or the <close-session> operation, discards the private candidate and all uncommitted configuration changes.

Locking private candidates in NETCONF

When using the private candidate mode over NETCONF, it is less likely that the candidate configuration needs to be locked. There can still be specific operational situations where the operator wants to use the private candidate functionality and temporarily restrict all other users from changing the node’s running configuration.

In this case, the operator can use the <lock> RPC as they would in the usual operating mode for NETCONF.

If an operator issues the <lock> RPC, the following applies:

- The private candidate configuration is promoted to a private-exclusive candidate configuration within SR OS.
- Any configuration changes that exist in the operator’s private candidate are retained.
- No other users on the SR OS node can commit any configuration changes.

When the operator who issued the <lock> RPC wants to allow others users to commit configuration changes again, the operator issues the <unlock> RPC.

If the operator issues the <unlock> RPC, the following applies:

- The other users get the rights to commit configuration changes again.
- Any uncommitted configuration changes in the private-exclusive candidate configuration are retained.
- The operator's private-exclusive candidate configuration is demoted back to private candidate status.

Note: In the default NETCONF mode (global candidate) and in the MD-CLI, uncommitted changes to an exclusive candidate configuration are discarded when the candidate is unlocked.

5.2.7 General NETCONF behavior

Use **Ctrl-c** in a NETCONF session to immediately terminate the session.

The SR OS NETCONF implementation supports XML namespaces (xmlns).

If an invalid namespace is specified within the client hello message, no error will be returned because the NETCONF server is still waiting for the client to send a valid <hello/>. For further NETCONF requests

(without sending a correct hello message), even though correct, SR OS returns a "Common base capability not found" error message.

SR OS checks for correct element namespaces on input and returns an error if they are incorrect.

An <edit-config> request must specify which data model (for example, Nokia SR OS YANG modules) is being used in the top-level <configure> element.

Example: Nokia SR OS namespace

The following example shows the Nokia SR OS namespace in the top level <configure> element.

```
<configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
   <system>
        ....
```

The NETCONF client can declare the namespaces with prefixes at the <rpc> tag and use the corresponding prefixes later in the request message <configure/> block. See Multiple use of standard NETCONF namespace for more information.

SR OS returns an error if the request contains one or more incorrect namespaces. See Invalid NETCONF namespace declaration for more information.

The chunked framing mechanism is supported in addition to the EOM mechanism. As described in RFC 6242, Section 4.1 - Framing Protocol, "[...] If the :base:1.1 capability is advertised by both peers, the chunked framing mechanism (see Section 4.2) is used for the remainder of the NETCONF session. Otherwise, the end-of-message-based mechanism (see Section 4.3) is used." See Chunked frame mechanism for more information.

Default data handling (for example, "info" vs "info detail") is supported in accordance with the mechanisms described in RFC 6243. The SR OS NETCONF server supports the 'explicit' method as the default mechanism for the Nokia SR OS YANG modules. The 'report-all' method is also supported for these modules.

The advertised capability changes depending on the YANG modules that are enabled or disabled in SR OS.

Example: Advertised capability

For example, when Nokia modules are enabled and all other modules are disabled, the following capability is advertised.

```
<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-
mode=explicit&amp;also-supported=report-all</capability>
```

Use the following command to dump NETCONF debug message streams:

- **MD-CLI**

  ```
  debug system management-interface netconf info
  ```

- **classic CLI**

  ```
  debug system netconf info
  ```

Note: In case of failure, the current logging levels do not mark the messages as errors or warnings.

5.2.7.1 Multiple use of standard NETCONF namespace

The following examples shows the standard NETCONF namespace "urn:ietf:params:xml:ns:netconf:base:1.0" used more than once in the <rpc> element.

Example: Standard NETCONF namespace used multiple times

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:alu="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get-config>
       <source> <running/> </source>
       <filter>
             <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                    <router>
                       <router-name>Base</router-name>
                       <interface>
                             <interface-name>system</interface-name>
                       </interface>
                    </router>
             </configure>
       </filter>
     </get-config>
</rpc>
]]>]]>
```

Example: Reply with an accepted namespace without error message

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
 xmlns:alu="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
            <router>
                <router-instance>Base</router-instance>
                <interface>
                    <interface-name>system</interface-name>
                    <admin-state>disable</admin-state>
                </interface>
            </router>
        </configure>
    </data>
</rpc-reply>
]]>]]>
```

5.2.7.2 Non-default NETCONF base namespace

Example: Allowed non-default NETCONF base namespace

The following example shows an allowed non-default NETCONF base namespace used in the <rpc> element.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:alu="urn:nokia.com:sros:ns:yang:sr:conf">
```

```
     <get-config>
          <source> <running/> </source>
          <filter>
               <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                  <router>
                       <router-name>Base</router-name>
                       <interface>
                             <interface-name>system</interface-name>
                       </interface>
                  </router>
               </configure>
          </filter>
     </get-config>
</rpc>
]]>]]>
```

Example: Output of the reply

In the following reply, a non-NETCONF base namespace is allowed and no error is returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:alu=="urn:nokia.com:sros:ns:yang:sr:conf">
    <data>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
            <router>
                <router-name>Base</router-name>
                <interface>
                    <interface-name>system</interface-name>
                    <admin-state>disable</admin-state>
                </interface>
            </router>
        </configure>
    </data>
</rpc-reply>
]]>]]>
```

5.2.7.3 Invalid NETCONF namespace declaration

Example: Invalid NETCONF namespace

The following example shows an invalid NETCONF namespace declared in the <rpc> element.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:alu="urn:alcatel-lucent.com:sros:ns:yang:sr:conf">
        <get-config>
                <source><running/></source>
                <filter>
                    <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                            <router>
                                    <router-name>Base</router-name>
                                    <interface>
                                         <interface-name>system</interface-name>
                                    </interface>
                            </router>
                    </configure>
                </filter>
```

```
        </get-config>
</rpc>
]]>]]>
```

Example: Output of the reply

In the following reply, SR OS returns an error.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
 xmlns:alu="urn:alcatel-lucent.com:sros:ns:yang:sr:conf">
    <rpc-error>
        <error-type>protocol</error-type>
        <error-tag>unknown-namespace</error-tag>
        <error-severity>error</error-severity>
        <error-message>
          An unexpected namespace is present.
        </error-message>
        <error-info>
          <bad-element>rpc</bad-element>
          <bad-namespace>urn:alcatel-lucent.com:sros:ns:yang:sr:conf</bad-namespace>
        </error-info>
    </rpc-error>
</rpc-reply>
]]>]]>
```

5.2.7.4 Non-default NETCONF namespace or prefix declaration in a child tag

Example: Non-default NETCONF namespace or prefix declaration

The following example shows a non-default NETCONF namespace or prefix declared in any child tag overriding the one declared under the <rpc> tag.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:alu="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get-config>
          <source> <running/> </source>
          <filter>
             <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                <router>
                    <router-name>Base</router-name>
                    <interface xmlns:alu="urn:nokia.com:sros:ns:yang:sr:conf">
                          <alu:interface-name>system</alu:interface-name>
                    </interface>
                </router>
             </configure>
          </filter>
     </get-config>
</rpc>
]]>]]>
```

Example: Output of the reply

In the following reply, the non-standard NETCONF namespace or prefix used in the tag is ignored.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
```

```
 xmlns:alu="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
            <router>
                <router-name>Base</router-name>
                <interface>
                    <interface-name>system</interface-name>
                    <admin-state>disable</admin-state>
                </interface>
            </router>
        </configure>
    </data>
</rpc-reply>
]]>]]>
```

5.2.7.5 Chunked frame mechanism

Example: Chunked message

```
#359
<?xml version="1.0" encoding="UTF-8"?><rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><get-config><source><running/></
source><filter>
<configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf"><router><router-name>Base</
router-name>
<interface><interface-name>system</interface-name></interface></router></
configure></filter></get-
config></rpc>
##
```

Example: Output of the reply

```
#38
<?xml version="1.0" encoding="UTF-8"?>
#1
#10
<rpc-reply
#17
 message-id="101"
#48
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
#1
>
#1
#9
    <data
#1
>
#1
#63
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
#21
            <router>
#48
                <router-name>Base</router-name>
#28
                <interface>
#60
                    <interface-name>system</interface-name>
#55
```

```
                    <admin-state>disable</admin-state>
#29
                </interface>
#22
            </router>
#21
        </configure>
#11
    </data>
#1
#12
</rpc-reply>
##
```

5.2.8 Establishing a NETCONF session

Example: Initiating a connection to a NETCONF server

The following example shows a client on a Linux PC initiating a connection to an SR OS NETCONF server. In accordance with RFC 6242, the SSH session must be invoked using an SSH subsystem.

```
ssh user_name@netconf_server_ip -p port_number -s netconf
```

The following examples show an exchange of hello messages that include advertisement of capabilities.

Example: Exchange of hello messages

The following is a message from the SR OS server.

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
        <capability>urn:ietf:params:netconf:base:1.0</capability>
        <capability>urn:ietf:params:netconf:base:1.1</capability>
        ...
        ...
    </capabilities>
    <session-id>20</session-id>
</hello>
]]>]]>
```

Example

A NETCONF client can reply with one of the following hello messages:

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
        <capability>urn:ietf:params:netconf:base:1.0</capability>
    </capabilities>
</hello>
]]>]]>
```

or

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
```

```
    <capability>urn:ietf:params:netconf:base:1.1</capability>
  </capabilities>
</hello>
]]>]]>
```

5.2.8.1 Checking NETCONF status

Example: <get-config> request on the <running> datastore

The following example shows a <get-config> request on the <running> datastore that checks on whether NETCONF is shut down or not on the router.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
          <source><running/></source>
          <filter>
               <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                       <system>
                            <management-interface>
                                   <netconf/>
                            </management-interface>
                       </system>
               </configure>
          </filter>
</get-config>
</rpc>
]]>]]>
```

Example: Output of the reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
            <system>
                <management-interface>
                   <netconf>
                      <admin-state>enable</admin-state>
                      <auto-config-save>true</auto-config-save>
                   </netconf>
                <management-interface>
            </system>
        </configure>
    </data>
</rpc-reply>
]]>]]>
```

5.2.8.2 Retrieving system configurations, QoS and log branches

Example: <get-config> request on the <candidate> datastore

The following example shows a <get-config> request on the <candidate> datastore to get the full configurations of the system, QoS, and log branches.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
<get-config>
    <source><candidate/></source>
    <filter>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
           <system>
           </system>
        </configure>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
            <log/>
        </configure>
    </filter>
</get-config>
</rpc>
]]>]]>
```

Example: Output of the reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
            <log>
                <filter>
                    <filter-id>1001</filter-id>
                    <entry>
                       <entry-id>10</entry-id>
                       <description>events of major severity or higher</description>
                       <action>forward</action>
                       <match>
                           <severity>
                               <gte>major</gte>
                           </severity>
                       </match>
                    </entry>
                </filter>
...
...
                <log-id>
                    <id>101</id>
                    <destination>
                        <netconf>
                        </netconf>
                    </destination>
                </log-id>
            </log>
            <system>
                <name>Test</name>
                <dns>
                    <address-pref>ipv4-only</address-pref>
                </dns>
...
...
            </system>
        </configure>
    </data>
</rpc-reply>
]]>]]>
```

5.2.8.3 Creating an Epipe service

Example: <edit-config> request on the <candidate> datastore

The following example shows an <edit-config> request on the <candidate> datastore to create a basic Epipe service.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
    <target><candidate/></target>
    <config>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
            <service>
                <epipe>
                    <service-name>CustDoc</service-name>
                    <customer>1</customer>
                    <service-mtu>1514</service-mtu>
                </epipe>
            </service>
        </configure>
    </config>
</edit-config>
</rpc>
]]>]]>
```

Example: Output of the reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

5.2.8.4 Returning multiple errors

The following examples show SR OS returning multiple errors with the <commit>.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
    <target><candidate/></target>
    <config>
        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
            <router>
                <router-name>Base</router-name>
                <ldp>
                    <interface-parameters>
                        <interface>
                            <ip-int-name>xe-1/1/1</ip-int-name>
                            <ipv4>
                            </ipv4>
                        </interface>
                        <interface>
                            <ip-int-name>xe-1/2/1</ip-int-name>
                            <ipv4>
```

```
                            </ipv4>
                        </interface>
                    </interface-parameters>
                    <targeted-session>
                        <peer>
                        <ip-address>172.22.1.34</ip-address>
                        </peer>
                    </targeted-session>
                    <tcp-session-parameters>
                        <peer-transport>
                        <ip-address>172.22.1.34</ip-address>
                        <authentication-key>Ru4bf!n</authentication-key>
                        </peer-transport>
                    </tcp-session-parameters>
                </ldp>
            </router>
         </configure>
      </config>
</edit-config>
</rpc>
]]>]]>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <commit/>
 </rpc>
]]>]]>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>operation-failed</error-tag>
        <error-severity>error</error-severity>
        <error-path xmlns:a="urn:nokia.com:sros:ns:yang:sr:conf">
/a:configure/a:router[a:router-name=&quot;Base&quot;]/a:ldp/a:interface-parameters/
a:interface[a:ip-int-name=&quot;xe-1/1/1&quot;]
        </error-path>
        <error-message>
            MINOR: MGMT_CORE #224: Entry does not exist - configure router router-
name &quot;Base&quot; interface interface-name
&quot;xe-1/1/1&quot;
        </error-message>
        <error-info>
            <err-element>interface</err-element>
        </error-info>
    </rpc-error>
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>operation-failed</error-tag>
        <error-severity>error</error-severity>
        <error-path xmlns:a="urn:nokia.com:sros:ns:yang:sr:conf">
/a:configure/a:router[a:router-name=&quot;Base&quot;]/a:ldp/a:interface-parameters/
a:interface[a:ip-int-name=&quot;xe-1/2/1&quot;]
        </error-path>
        <error-message>
```

```
            MINOR: MGMT_CORE #224: Entry does not exist - configure router router-
name &amp;quot;Base&amp;quot; interface interface-name
&amp;quot;xe-1/2/1&amp;quot;
        </error-message>
        <error-info>
            <err-element>interface</err-element>
        </error-info>
    </rpc-error>
</rpc-reply>
]]>]]>
```

5.3 NETCONF notifications

NETCONF notifications support is a standard IETF asynchronous notification delivery service for NETCONF that is specified in RFC 5277. SR OS allows log events to be output as NETCONF notifications. NETCONF is one of the output options for an event log (along with SNMP, syslog, and others).

The :notification and :interleave capabilities are advertised in the SR OS NETCONF server <hello> using the following syntax.

```
<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
```

The following characteristics of the NETCONF notifications capabilities are supported in the SR OS:

- The :notification capability indicates that the SR OS NETCONF server can process a subscription and send event notifications to the NETCONF client.
- The :interleave capability indicates that the SR OS NETCONF server supports receiving, processing, and responding to NETCONF requests on the same NETCONF session that has an active notification subscription.
- A NETCONF client must maintain an open NETCONF session with the NETCONF server to receive NETCONF notifications.
- A NETCONF client can send a <create-subscription> RPC to the SR OS NETCONF server to start receiving notification messages.
- If the SR OS NETCONF server can satisfy the request, SR OS sends an <OK> element within the <rpc-reply>.
- If the SR OS NETCONF server cannot satisfy the request, SR OS sends an <rpc-error> element within the <rpc-reply>.
- Subscriptions are nonpersistent and their lifetime is defined by their NETCONF session (subscriptions are not maintained when a router reboots).
- An optional [stream] parameter can be defined for a <create-subscription> RPC. The following are characteristics of the [stream] parameter.
  - An event stream is a set of event notifications matching a specified forwarding criteria and available to the NETCONF clients for subscription.
  - A NETCONF session can subscribe to only one stream at a time.
  - One stream can be subscribed-to by many NETCONF sessions.
  - The SR OS NETCONF server maintains one or more event streams.
  - SR OS uses the SR OS event reporting framework for NETCONF notifications.

- A log-id can be configured to be a NETCONF stream. A "netconf-stream" exists per log-id. It is used to assign a NETCONF "stream" name with a log-id. A "netconf-stream" is unique per SR OS device. It must be configured with "to netconf" for subscriptions to be accepted. If a "netconf-stream" is changed, active subscriptions to the changed NETCONF stream name are terminated by the SR OS.
- Log-id 101 is the default, preconfigured stream with the "netconf-stream" set to "NETCONF". This stream is used by default if the [stream] parameter is not specified. The preconfigured stream is modifiable but not deletable.
- Other streams can be configured using NETCONF or CLI. These streams are user-configured; are modifiable and deletable. Because "NETCONF" is reserved for the preconfigured stream (that is, log-id 101), a user-configured "netconf-stream" cannot be set to "NETCONF".
- When a NETCONF client tries to subscribe to the SNMP log-id or a non-configured log-id, SR OS returns an error.
- SR OS supports up to 64 concurrent subscriptions to all streams.
- Notifications can be filtered out using a log-id "filter" or using base-op for create-subscriptions RPC.
- After the NETCONF server receives an SR OS event through a stream, a <notification> element is ready to be sent to all NETCONF sessions subscribed to that stream as per their filters.

• SR OS maps log events to the following NETCONF notifications. See NETCONF notification examples for more information.

- **sros-config-change-event**

  This notification contains information about configuration changes in classic format. The tmnxConfigModify, tmnxConfigCreate, and tmnxConfigDelete log events from the SYSTEM and SECURITY applications are mapped to this notification. For model-driven information about configuration changes, Nokia recommends you use ON_CHANGE telemetry (for example, a subscription to the YANG /configure path) or the netconf-config-change notification that follows.

- **sros-state-change-event**

  This notification contains information about state changes in classic format. The tmnxStateChange log event from the SYSTEM and SECURITY applications are mapped to this notification. For model-driven information about state changes, Nokia recommends you use ON_CHANGE telemetry (for example, a subscription to various YANG state paths).

- **sros-command-accounting-event**

  This notification contains information about the commands and operations performed by users in classic CLI and MD-CLI. Log events mapped to this notification include the cli_*_io events from the USER application and the md_cli_io and md_cli_unauth_io events from the SECURITY application.

- **sros-log-generic-event**

  This notification contains information about SR OS events in most protocols and feature areas. All log events that are not mapped to any other notification in this list are mapped to the **sros-log-generic-event** notification.

- **netconf-config-change**

  This is a notification based on the model-driven configuration change log events "mdConfigChange", "mdOcConfigChange", "mdBofConfigChange", and "mdDebugConfigChange" from the MGMT_CORE application. The notification is sent upon any configuration change that occurs in the running datastore by a model-driven management interface, using either the Nokia SR OS or OpenConfig data models, and in any configuration region except li (such as configure, bof,

and debug). By default, the notification is disabled because all corresponding log events are also disabled by default. The notification uses the standard notification: netconf-config-change (as per RFC 6470) augmented with a value leaf.

A single configuration change may involve editing more than one object (target). Each "mdConfigChange" log event contains only a single object edit. As a result, only one object (target) edit can exist per **netconf-config-change** notification. Bundling of edits in a single **netconf-config-change** notification does not occur.

- **sros-md-rpc-accounting-event**

  This notification is based on the NETCONF/gRPC local command accounting log events (the netconf_auth, netconf_unauth, grpc_auth, and grpc_unauth log events from the SECURITY application). This notification is sent upon receiving any RPC from a NETCONF/gRPC client. The NETCONF/gRPC local command accounting log events and NETCONF notification do not show the details of the configuration changes sent via the NETCONF/gRPC RPCs.

- SR OS supports the following Lawful Intercept (LI) NETCONF notifications.
  - **sros-li-config-change-event**

    This notification contains information about LI configuration changes in classic format. The tmnxConfigModify, tmnxConfigCreate, and tmnxConfigDelete LI log events from the LI application are mapped to this notification. For model-driven information about LI configuration changes, Nokia recommends you use the netconf-li-config-change notification described in the following bullets.

  - **sros-li-state-change-event**

    This notification contains information about LI state changes in classic format. The tmnxStateChange log event from the LI application is mapped to this notification.

  - **sros-li-command-accounting-event**

    This notification contains information about which LI user performed which commands and operations in classic CLI and MD-CLI. LI log events mapped to this notification include the cli_*_io events from the LI application as well as the md_cli_io and md_cli_unauth_io events from the LI application.

  - **sros-li-log-generic-event**

    This notification contains information about LI SR OS log events in most protocols and feature areas. All LI SR OS log events that are not mapped to any other notifications in this list are mapped to the **sros-li-log-generic-event**.

  - **netconf-li-config-change**

    This is a notification based on the model-driven LI configuration change log event (the "mdLiConfigChange" log event from the LI application). It is sent upon any LI configuration change that occurs in the running datastore by a model-driven management interface. By default, this notification is enabled, because the log event is also enabled by default.

  - **sros-md-li-rpc-accounting-event**

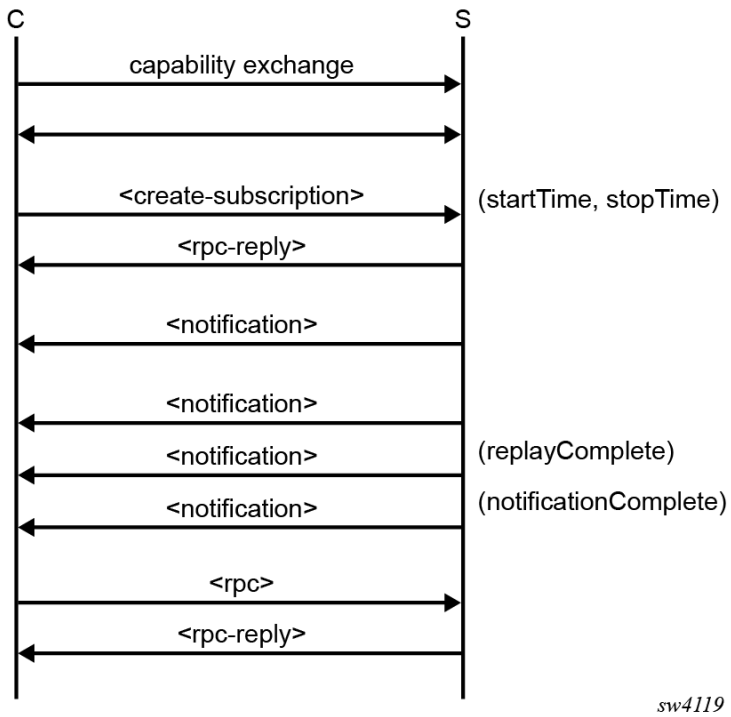
    This is a notification based on the NETCONF/gRPC local command accounting LI log events (the netconf_auth, netconf_unauth, grpc_auth, and grpc_unauth log events from the LI application). This notification is sent upon receiving any RPC from a NETCONF/gRPC client. The NETCONF/gRPC local command accounting LI log events and LI NETCONF notification provide basic information about which RPC/operation was requested and the associated user. They do not show the details of every configuration element changed in a NETCONF request.

The following are the characteristics of a <create-subscription> RPC:

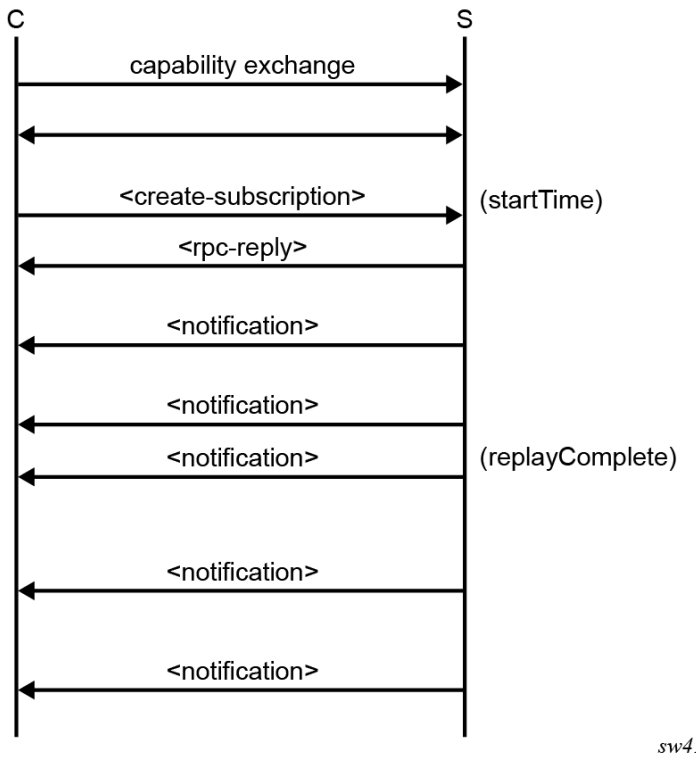
- The <filter> argument is optional and is not supported by SR OS.
- The <startTime> optional argument triggers the starting time of a replay. If it is not present, the subscription cannot be used to replay. The <startTime> argument cannot specify a time that is later than the current time (that is, in the future). SR OS supports time zones.
- The <stopTime> optional argument triggers the stop time. If it is not present, notifications continue to be sent until the subscription is terminated. The <stopTime> argument can specify a time that is later than the current time (that is, in the future). SR OS supports time zones.

A replay buffer is maintained by the SR OS server (per stream) and sorted by the order they were initially sent out (that is, by sequence-id, and not by timestamps). The following are the characteristics of replay requests:

- A replay request from the client causes stored events to be sent to the client for the specified time interval.
- A stream that supports replay is not expected to have an unlimited supply of saved notifications available to accommodate any replay request.
- The <startTime> and <stopTime> arguments are used to specify when collections begin and end, respectively.
- A <replayComplete> notification is sent to indicate that all the replay notifications have been sent.
  - If a <stopTime> was specified, the session then becomes a normal NETCONF session, and the NETCONF server accepts <rpc> operations. A <notificationComplete> notification is expected after the <replayComplete> if a <stopTime> was specified. The following is an example of a session with a <stopTime> specified:



  - If a <stopTime> is not specified, the session continues to send notifications as they arise in the system. The following is an example of a session without a <stopTime> specified:



- If neither <startTime> nor <stopTime> arguments are present, no replay is present and notifications continue to be sent until the subscription is terminated.

5.3.1 NETCONF notification examples

This section provides examples of NETCONF notifications.

5.3.1.1 <create-subscription> operation

Example: Output of a <create-subscription> operation

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"/>
</rpc>
]]>]]>
```

Example: Output of a reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

5.3.1.2 sros-config-change-event notification

Example: Output of an sros-config-change-event notification

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2017-06-12T09:12:43.376Z</eventTime>
    <sros-config-change-event xmlns="urn:nokia.com:sros:ns:yang:sr:notifications">
        <sequence-number>8447</sequence-number>
        <severity>warning</severity>
        <application>system</application>
        <event-id>2008</event-id>
        <event-name>tmnxConfigDelete</event-name>
        <router-name>Base</router-name>
        <subject>LDP</subject>
        <message>vRtrLdpNgSessionTable: Virtual Router 1, Peer 2.2.2.2:0. managed ob
ject deleted</message>
        <event-params>
            <tmnxNotifyRow>vRtrLdpNgSessState.1.1.6.2.2.2.2.0.0</tmnxNotifyRow>
            <tmnxNotifyEntryOID>vRtrLdpNgSessionEntry</tmnxNotifyEntryOID>
            <tmnxNotifyObjectName>vRtrLdpNgSessionTable: Virtual Router 1, Peer 2.2.
2.2:0.</tmnxNotifyObjectName>
        </event-params>
    </sros-config-change-event>
</notification>
```

5.3.1.3 sros-state-change-event notification

Example: Output of an sros-state-change-event notification

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2017-06-12T09:16:36.781Z</eventTime>
    <sros-state-change-event xmlns="urn:nokia.com:sros:ns:yang:sr:notifications">
        <sequence-number>8460</sequence-number>
        <severity>warning</severity>
        <application>system</application>
        <event-id>2009</event-id>
        <event-name>tmnxStateChange</event-name>
        <router-name>Base</router-name>
        <subject>LDP</subject>
        <message>Status of vRtrLdpNgSessionTable: Virtual Router 1, Peer 2.2.2.2:0.
changed administrative state: inService, operational state: inService</message>
        <event-params>
            <tmnxNotifyRow>vRtrLdpNgSessState.1.1.6.2.2.2.2.0.0</tmnxNotifyRow>
            <tmnxNotifyRowAdminState>inService</tmnxNotifyRowAdminState>
            <tmnxNotifyRowOperState>inService</tmnxNotifyRowOperState>
            <tmnxNotifyEntryOID>vRtrLdpNgSessionEntry</tmnxNotifyEntryOID>
            <tmnxNotifyObjectName>vRtrLdpNgSessionTable: Virtual Router 1, Peer 2.2.
2.2:0.</tmnxNotifyObjectName>
        </event-params>
    </sros-state-change-event>
</notification>
```

5.3.1.4 sros-cli-accounting-event notification

Example: Output of an sros-cli-accounting-event notification

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2017-06-12T09:11:45.476Z</eventTime>
    <sros-command-accounting-
event mlns="urn:nokia.com:sros:ns:yang:sr:notifications">
        <sequence-number>8462</sequence-number>
        <severity>minor</severity>
        <application>user</application>
        <event-id>2011</event-id>
        <event-name>cli_config_io</event-name>
        <router-name>Base</router-name>
        <subject>admin</subject>
        <message>User from CONSOLE: Dut-C>config>log>log-id#  /
configure router interface "toDutB_214" </message>
        <event-params>
            <srcAddr>CONSOLE</srcAddr>
            <prompt>Dut-C>config>log>log-id# </prompt>
            <message>/configure router interface "toDutB_214" </message>
        </event-params>
    </sros-command-accounting-event>
</notification>
```

5.3.1.5 sros-log-generic-event notification

Example: Output of an sros-log-generic-event notification

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2017-06-12T09:12:42.344Z</eventTime>
    <sros-log-generic-event xmlns="urn:nokia.com:sros:ns:yang:sr:notifications">
        <sequence-number>8443</sequence-number>
        <severity>warning</severity>
        <application>ospf</application>
        <event-id>2047</event-id>
        <event-name>tmnxOspfNgIfStateChange</event-name>
        <router-name>Base</router-name>
        <subject>VR:  1 OSPFv2 (0) </subject>
        <message>LCL_RTR_ID 1.1.1.1: Interface toDutB_214 state changed to down (eve
nt IF_DOWN)</message>
        <event-params>
            <vRtrID>1</vRtrID>
            <tmnxOspfVersion>version2</tmnxOspfVersion>
            <tmnxOspfInstance>0</tmnxOspfInstance>
            <tmnxOspfRouterId>16843009</tmnxOspfRouterId>
            <tmnxOspfNgIfIndex>0x00000007</tmnxOspfNgIfIndex>
            <tmnxOspfNgIfInstId>0</tmnxOspfNgIfInstId>
            <tmnxOspfNgIfAreaId>0</tmnxOspfNgIfAreaId>
            <tmnxOspfNgIfState>down</tmnxOspfNgIfState>
            <tmnxOspfIfIpName>toDutB_214</tmnxOspfIfIpName>
            <tmnxOspfIfEvent>IF_DOWN</tmnxOspfIfEvent>
            <ospfRouterIdIpAddr>1.1.1.1</ospfRouterIdIpAddr>
        </event-params>
    </sros-log-generic-event>
</notification>
```

5.3.1.6 netconf-config-change notification

Example: Output of a netconf-config-change notification

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"> <eventTime>20
16-01-01T19:17:33Z</eventTime>
<netconf-config-change
      xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-notifications"
      xmlns:notif="urn:nokia.com:sros:ns:yang:sr:notifications"
      xmlns:sros="urn:nokia.com:sros:ns:yang:sr:conf">
      <changed-by>
        <username>user_name</username>
        <session-id>8</session-id>
        <source-host>138.192.72.45</remote-host>
      </changed-by>
    <datastore>running</datastore>
    <edit>
      <target>/config/service/epipe[serviceId=1]</target>
      <operation>create</operation>
      <notif:value>anyValue</notif:value>
    </edit>
</netconf-config-change>
</notification>
```

5.3.1.7 sros-md-rpc-accounting-event notification

Example: Output of an sros-md-rpc-accounting-event notification

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2019-10-08T21:01:50.165Z</eventTime>
    <sros-md-rpc-accounting- event xmlns="urn:nokia.com:sros:ns:yang:sr:notifications">
        <sequence-number>124</sequence-number>
        <severity>minor</severity>
        <application>security</application>
        <event-id>2227</event-id>
        <event-name>netconf_auth</event-name>
        <router-name>management</router-name>
        <subject>admin</subject>
        <message>User admin from 192.168.7.229 port 44559 to port 830 session 7: edi
t-config RPC authorized</message>
        <event-params>
            <userName>admin</userName>
            <srcAddr>192.168.7.229</srcAddr>
            <srcPort>44559</srcPort>
            <dstPort>830</dstPort>
            <sessionId>7</sessionId>
            <rpcName>edit-config</rpcName>
        </event-params>
    </sros-md-rpc-accounting-event>
</notification>
]]>]]>
```

5.4 NETCONF monitoring

The :ietf-netconf-monitoring capability is advertised in the SR OS NETCONF server <hello> using the following syntax.

```
<capability>urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring</capability>
```

The advertised capability provides information about the schemas supported by SR OS, which allows a NETCONF client to query and retrieve schema information from the SR OS NETCONF server.

SR OS supports all subtrees from the YANG model that is used to monitor the NETCONF protocol as described in RFC 6022 (that is, ":ietf-netconf-monitoring" capability).

A <get-schema> operation is supported for explicit schema retrieval using NETCONF (YANG data model discovery and download as described in RFC 6022). The following parameters are supported:

- **identifier**

  This parameter is a mandatory string. It specifies an identifier for the schema list entry (YANG file). It can be the name of a module or a submodule.

- **version**

  This parameter is an optional string. It specifies a version of the schema requested (for example, YANG file). It represents the most recent YANG **revision** statement in a module or submodule. There is an empty string if no **revision** statement is present. Because multiple versions may be supported by the NETCONF server, each version must be reported individually in the schema list (it can have same identifier but different versions).

- **format**

  This parameter is an optional string. It specifies the data modeling language in which the schema is written. The default value is **yang** when not specified. If specified, **yang** is the only value supported.

Unless the user intentionally specifies a schema path destination to acquire the YANG schema files, the software upgrade process manages the YANG schema files to ensure the schema files are synchronized with the software image on both the primary and standby CPM.

When an SR OS image boots from the primary image, the associated YANG files match the image. If the primary SR OS image fails to boot, and the secondary or tertiary SR OS image loads, the YANG schema files associated with the loaded image are installed and available to the <get-schema> NETCONF RPC. The YANG files are delivered with the software image as part of the `yang.tim` file. Nokia recommends that the primary, secondary, and tertiary image strings should not exceed 120 characters each for the <get-schema> request to work correctly with all schema files.

Use the following commands to configure the BOF image:

- **MD-CLI**

  ```
  bof image primary-location
  bof image secondary-location
  bof image tertiary-location
  ```

- **classic CLI**

  ```
  bof primary-image
  bof secondary-image
  ```

```
bof tertiary-image
```

Use the following command to configure the schema path.

```
configure system management-interface schema-path
```

For the MD-CLI, see *7705 SAR Gen 2 MD-CLI Command Reference Guide* for more information about the **schema-path** command.

For the classic CLI, see the *7705 SAR Gen 2 Classic CLI Command Reference Guide* for more information about the **schema-path** command.

If this command is configured, all YANG schema files must be manually copied to the specified schema path. The directory structure and path of the schema files must be maintained, so that the <get-schema> RPC is successful.

When the requested schema does not exist, the "invalid-value" <error-tag> is returned. The maximum length of a schema path is 180 characters. However, Nokia recommends that a specified schema path be less than or equal to 135 characters, to guarantee that a <get-schema> works properly with the longest YANG module name in SR OS.

Example: Output of a <get-schema> request

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-schema xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
        <identifier>nokia-conf</identifier>
    </get-schema>
</rpc>
]]>]]>
```

Example: Output of a reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-
monitoring"><![CDATA[module nokia-conf {
    yang-version "1.1";
    namespace "urn:nokia.com:sros:ns:yang:sr:conf";
..
}
]]></data>
</rpc-reply>
```

Use the following command option to synchronize the `yang.tim` file associated with each image between the primary and standby CPM:

- **MD-CLI**

```
admin redundancy synchronize boot-environment
```

- **classic CLI**

```
admin redundancy synchronize boot-env
```

If model-driven management is not enabled on the device and disk space is limited, the `yang.tim` file can be deleted from the flash card. If the file is missing, the system still successfully completes the synchronization, but displays a warning message and reports that it failed to copy the file in the system

logs. Nokia recommends storing the `yang.tim` file with the other software image components on the compact flash.

If the schema path is configured to be a local directory, the preceding command recursively copies all YANG files from the local directory to the standby CPM.

5.4.1 netconf-state schemas

The schemas subtree provides a list of the supported YANG schemas on the SR OS device.

The data returned for a query of the `/netconf-state/schemas` path depends on the settings of the various commands in the **yang-modules** container. Examples are available at the end of this section.

A `/netconf-state/schemas` path returns all supported YANG modules.

Example: Output of a request and the received response

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
      <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
        <schemas/>
      </netconf-state>
    </filter>
  </get>
</rpc>
]]>]]>
```

Example: Output of the reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
            <schemas>
                <schema>
                    <identifier>nokia-conf</identifier>
                    <version>2016-07-06</version>
                    <format>yang</format>
                    <namespace>urn:nokia.com:sros:ns:yang:sr:conf</namespace>
                    <location>NETCONF</location>
                </schema>
                <schema>
                    <identifier>nokia-conf-aa-group</identifier>
                    <version>2018-09-14</version>
                    <format>yang</format>
                    <namespace>urn:nokia.com:sros:ns:yang:sr:conf</namespace>
                    <location>NETCONF</location>
                </schema>
                <schema>
                    <identifier>nokia-conf-aaa</identifier>
                    <version>2018-08-27</version>
                    <format>yang</format>
                    <namespace>urn:nokia.com:sros:ns:yang:sr:conf</namespace>
                    <location>NETCONF</location>
                </schema>
                    ….
                    ….
                <schema>
```

```
                    <identifier>nokia-state</identifier>
                    <version>2016-07-06</version>
                    <format>yang</format>
                    <namespace>urn:nokia.com:sros:ns:yang:sr:state</namespace>
                    <location>NETCONF</location>
                </schema>
                <schema>
                    <identifier>nokia-state-aa-group</identifier>
                    <version>2018-09-14</version>
                    <format>yang</format>
                    <namespace>urn:nokia.com:sros:ns:yang:sr:state</namespace>
                    <location>NETCONF</location>
                </schema>
                <schema>
                    <identifier>nokia-state-aaa</identifier>
                    <version>2018-08-27</version>
                    <format>yang</format>
                    <namespace>urn:nokia.com:sros:ns:yang:sr:state</namespace>
                    <location>NETCONF</location>
                </schema>
                    …
                    …
      </schemas>
    </state>
  </data>
</rpc-reply>
```

5.4.2 netconf-state datastores

The datastores subtree provides a list of configuration datastores supported on the SR OS device.

The data returned for a query of the `/netconf-state/datastores` path depends on the settings of the **yang-modules** container and whether NMDA support is configured.

Example: netconf-state datastores request and received response

```
<rpc message-id="50" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
        <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
            <datastores/>
        </netconf-state>
    </filter>
  </get>
</rpc>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="50" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
            <datastores>
                <datastore>
                    <name>running</name>
                </datastore>
                <datastore>
                    <name>candidate</name>
                </datastore>
                <datastore>
                    <name>startup</name>
                </datastore>
                [..SNIP..]
```

```
                </datastore>
            </datastores>
        </netconf-state>
    </data>
</rpc-reply>
]]>]]>
```

5.4.3 netconf-state sessions

The sessions subtree provides a list of all active NETCONF sessions.

Example: netconf-state session request and received response

```
<rpc message-id="50" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
        <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
            <sessions/>
        </netconf-state>
    </filter>
  </get>
</rpc>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="50" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
            <sessions>
                <session>
                    <session-id>249</session-id>
                    <transport xmlns:ietf-netconf-monitoring=
"urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">ietf-netconf-monitoring:netconf-
ssh</transport>
                    <username>admin</username>
                    <source-host>192.168.168.2</source-host>
                    <login-time>2022-06-23T13:11:08.0Z</login-time>
                    <in-rpcs>1</in-rpcs>
                    <in-bad-rpcs>0</in-bad-rpcs>
                    <out-rpc-errors>0</out-rpc-errors>
                    <out-notifications>0</out-notifications>
                </session>
            </sessions>
        </netconf-state>
    </data>
</rpc-reply>
]]>]]>
```

5.4.4 netconf-state capabilities

The capabilities subtree provides a list of the capabilities of the current NETCONF session. The output is similar to the NETCONF hello message.

5.4.5 netconf-state statistics

The statistics subtree provides statistical data pertaining to the NETCONF service on the SR OS device.

Example: netconf-state statistics request and received response

```
<rpc message-id="50" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
        <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
            <statistics/>
        </netconf-state>
    </filter>
  </get>
</rpc>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="50" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
            <statistics>
                <netconf-start-time>2022-06-22T20:22:21.0Z</netconf-start-time>
                <in-sessions>2</in-sessions>
                <in-rpcs>4</in-rpcs>
                <in-bad-rpcs>1</in-bad-rpcs>
                <out-rpc-errors>1</out-rpc-errors>
                <out-notifications>0</out-notifications>
            </statistics>
        </netconf-state>
    </data>
</rpc-reply>
]]>]]>
```

5.5 YANG library

SR OS supports the YANG library mechanism to identify the YANG modules and submodules that are implemented by the NETCONF server. NETCONF clients should be able to query or cache the YANG library contents and identify whether their cache is out of date.

The SR OS NETCONF server supports the "/yang-library" state model and advertises the following capability in the <hello> message (in accordance with RFC 8525).

```
<capability>urn:ietf:params:netconf:capability:yang-
library:1.1?revision=<date>&content-id=<content-id-value></capability>
```

The following is the YANG tree diagram for the "/yang-library" model.

```
 module: ietf-yang-library
     +--ro yang-library
        +--ro module-set* [name]
        |  +--ro name                  string
        |  +--ro module* [name]
        |  |  +--ro name         yang:yang-identifier
        |  |  +--ro revision?    revision-identifier
        |  |  +--ro namespace    inet:uri
        |  |  +--ro location*    inet:uri
        |  |  +--ro submodule* [name]
        |  |  |  +--ro name        yang:yang-identifier
        |  |  |  +--ro revision?   revision-identifier
        |  |  |  +--ro location*   inet:uri
        |  |  +--ro feature*     yang:yang-identifier
        |  |  +--ro deviation*   -> ../../module/name
```

```
        |  +--ro import-only-module* [name revision]
        |     +--ro name         yang:yang-identifier
        |     +--ro revision     union
        |     +--ro namespace    inet:uri
        |     +--ro location*    inet:uri
        |     +--ro submodule* [name]
        |        +--ro name        yang:yang-identifier
        |        +--ro revision?   revision-identifier
        |        +--ro location*   inet:uri
        +--ro schema* [name]
        |  +--ro name          string
        |  +--ro module-set*   -> ../../module-set/name
        +--ro datastore* [name]
        |  +--ro name      ds:datastore-ref
        |  +--ro schema    -> ../../schema/name
        +--ro content-id    string
```

The SR OS NETCONF server advertises the following capability in the <hello> message:

```
<capability>urn:ietf:params:netconf:capability:yang-library:1.0?revision=<revision-
date>&amp;module-set-id=<string></capability>
```

The following is the YANG tree diagram for the **modules-state** model:

```
 +--ro modules-state
   +--ro module-set-id    string
   +--ro module* [name revision]
     +--ro name                yang:yang-identifier
     +--ro revision            union
     +--ro schema?             inet:uri
     +--ro namespace           inet:uri
     +--ro feature*            yang:yang-identifier
     +--ro deviation* [name revision]
     |  +--ro name        yang:yang-identifier
     |  +--ro revision    union
     +--ro conformance-type    enumeration
     +--ro submodule* [name revision]
       +--ro name        yang:yang-identifier
       +--ro revision    union
       +--ro schema?     inet:uri
```

The **module-set-id** is a mandatory leaf that identifies a set of YANG modules that the SR OS NETCONF server supports. The value of this leaf changes whenever there is a change in the set of modules or submodules in the YANG library. When this change occurs, SR OS changes the **module-set-id** value advertised in the NETCONF server <hello> message.

The NETCONF client can use the **modules-state** to fetch the YANG library, cache it, and re-fetch it only if the value of the **module-set-id** changes again. The YANG library is returned in the **module** list.

If the SR OS NETCONF server advertises the following capability, the NETCONF client can use the advertised **module-set-id** to query the YANG library.

```
    <capability>urn:ietf:params:netconf:capability:yang-library:1.0?revision=2018-
    05-08&module-set-id=1234</capability>
```

Example: Output of the NETCONF client query

```
    <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
    <get>
     <filter type="subtree">
```

```
      <modules-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
         <module-set-id>1234</module-set-id>
         <module>
         </module>
       </modules-state>
     </filter>
   </get>
  </rpc>
  ]]>]]>
```

Example: Output of the reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <modules-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
            <module-set-id>1234</module-set-id>
            <module>
                <name>iana-if-type</name>
                <revision>2014-05-08</revision>
                <namespace>urn:ietf:params:xml:ns:yang:iana-if-type</namespace>
                <conformance-type>implement</conformance-type>
            </module>
                        …
                        …
            <module>
                <name>nokia-conf</name>
                <revision>2016-07-06</revision>
                <namespace>urn:nokia.com:sros:ns:yang:sr:conf</namespace>
                <conformance-type>implement</conformance-type>
                <submodule>
                    <name>nokia-conf-aa-common</name>
                   <revision>2018-04-23</revision>
                </submodule>
                                …
                                …
            </module>
                        …
                        …
        </modules-state>
    </data>
</rpc-reply>
]]>]]>
```

5.6 Operational commands via NETCONF

Operational commands (for example, **admin reboot**, **file remove**) are supported via NETCONF using:

- individually YANG-modeled operations
- md-cli-raw-command

Nokia recommends using an individually modeled YANG operation, if it is available for a specific operation, instead of the md-cli-raw-command. The individually YANG-modeled operations are useful for operations that have specific results data in the output (providing fully modeled and structured output). The md-cli-raw-command is available for all SR OS operations but requires string parsing of the output for commands with results data.

Individually modeled operations that are supported over NETCONF can be viewed in the `nokia-oper-*.yang` files in the YANG distribution of a specific software release.

5.6.1 Individually YANG-modeled operations

Some operations are modeled using a YANG "action" statement per operation. These can be viewed in the `nokia-oper-*.yang` files.

The commands in the YANG operations models are invoked via NETCONF using the "action" RPC, as described in section 7.15 of RFC 7950.

For more information, see YANG-modeled operations.

Example: Output of a request to create a new directory on CF1

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <action xmlns="urn:ietf:params:xml:ns:yang:1">
        <file xmlns="urn:nokia.com:sros:ns:yang:sr:oper-file">
            <make-directory>
                <url>cf1:\\my-folder-abc</url>
            </make-directory>
         </file>
    </action>
</rpc>
]]>]]>
```

Example: Output of a reply for a successful directory creation operation

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:nokiaoper="urn:nokia.com:sros:ns:yang:sr:oper-file">
    <nokiaoper:operation-id>10</nokiaoper:operation-id>
    <nokiaoper:start-time>2021-06-16T20:09:00.7Z</nokiaoper:start-time>
    <nokiaoper:status>completed</nokiaoper:status>
    <nokiaoper:end-time>2021-06-16T20:09:06.7Z</nokiaoper:end-time>
</rpc-reply>
]]>]]>
```

Example: Output of a CPM synchronize operation request

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <admin xmlns="urn:nokia.com:sros:ns:yang:sr:oper-admin">
        <redundancy>
            <synchronize>
                <configuration/>
            </synchronize>
        </redundancy>
    </admin>
  </action>
</rpc>
]]>]]>
```

Example: Output of a reply when the CPM synchronize operation fails

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>operation-failed</error-tag>
        <error-severity>error</error-severity>
        <error-message>
            MINOR: MGMT_AGENT #2007: Operation failed - secondary CPM offline
        </error-message>
    </rpc-error>
</rpc-reply>
]]>]]>
```

5.6.1.1 <md-compare> YANG-modeled operation

NETCONF can be used to obtain the differences between configurations. The <md-compare> operation is a synchronous YANG modelled operation. See Individually YANG-modeled operations for more information.

The YANG model with detailed input and output parameters for the <md-compare> operation is available in the YANG distribution for the specific software release.

The following table summarizes a selection of the available input parameters.

Table 29: <md-compare> input parameters

Input parameter	Description
<path>/<subtree-path>	The XML-formatted and correctly namespaced path to the tree location where the <md-compare> operation is executed. Omitting the <subtree-path> executes the <md-compare> operation from the root of the YANG tree. Optional. Default: root (/)
<configuration-region>	The SR OS configuration region. See <get> for an example of the <configuration-region>. Optional. Default: configure
<format>	The NETCONF output format. See Output format selection for more information. Optional. Default: xml
<source>	The source configuration. See Table 30: Source and destination locations for more information. Optional. Default: <candidate/>
<destination>	The destination configuration. See Table 30: Source and destination locations for more information. Optional. Default: <baseline/>

The following table describes the source and destination locations available for use.

Table 30: Source and destination locations

Option	Description
<candidate/>	The candidate configuration. This is the default for the <source> input parameter.
<baseline/>	The baseline configuration. This is the default for the <destination> input parameter.
<url>URL</url>	Uses the location specified in the URL. The following URL types are supported: • local (`cf1:`, `cf2:`, `cf3:`, and so on) • non-active CPMs (`cf1-b:`, `cf2-b:`, and so on) • remote FTP or TFTP locations
<running/>	The running configuration
<startup/>	The startup configuration
<booted/>	The booted configuration. To use this option, the <configuration-region> must be set to "bof".
<rollback-id>NUMBER</rollback>	The configuration saved a NUMBER of times ago
<commit-id>NUMBER</commit-id>	The configuration at the commit-id identified by NUMBER

The results from the <md-compare> operation are displayed in the <results>/<md-compare-output> XML field.

The **xml** formatted <md-compare> provides the correctly formatted and namespace-aware XML data that, if sent to the node, results in the target configuration. This option is similar to the use of the following command in the MD-CLI.

```
compare summary netconf-rpc
```

The **md-cli** formatted <md-compare> provides the equivalent output as the following command in the MD-CLI.

```
compare summary
```

The **md-cli-full** formatted <md-compare> provides the equivalent output as the following command in the MD-CLI.

```
compare
```

Output example: <md-compare> configuration

```
<rpc message-id="2" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <edit-config>
        <target><candidate/></target>
        <config>
```

```
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                <log>
                    <log-id>
                        <name>50</name>
                        <admin-state>enable</admin-state>
                        <description>Example</description>
                        <source>
                            <main>true</main>
                            <change>true</change>
                        </source>
                        <destination>
                            <cli>
                            </cli>
                        </destination>
                    </log-id>
                </log>
            </configure>
        </config>
    </edit-config>
</rpc>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="2" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <ok/>
</rpc-reply>
]]>]]>
```

See the following sections for examples of the <md-compare> operation:

- <md-compare> using all default values
- <md-compare> using MD-CLI format
- <md-compare> using XML format and an explicit path

5.6.1.1.1 <md-compare> using all default values

Example

```
<rpc message-id="121" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <action xmlns="urn:ietf:params:xml:ns:yang:1">
        <global-operations xmlns="urn:nokia.com:sros:ns:yang:sr:oper-global">
            <md-compare/>
        </global-operations>
    </action>
</rpc>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="121" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1" xmlns:nokiaoper="urn:nokia.com:sros:ns:yang:sr:oper-
global">
    <nokiaoper:operation-id>11</nokiaoper:operation-id>
    <nokiaoper:start-time>2022-06-20T22:03:09.9Z</nokiaoper:start-time>
    <nokiaoper:results>
        <nokiaoper:md-compare-output>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
                <log>
                    <log-id>
                        <name>50</name>
```

```
                        <admin-state>enable</admin-state>
                        <description>Example</description>
                        <source>
                            <main>true</main>
                            <change>true</change>
                        </source>
                        <destination>
                            <cli>
                            </cli>
                        </destination>
                    </log-id>
                </log>
            </configure>
        </nokiaoper:md-compare-output>
    </nokiaoper:results>
    <nokiaoper:status>completed</nokiaoper:status>
    <nokiaoper:end-time>2022-06-20T22:03:10.0Z</nokiaoper:end-time>
</rpc-reply>
]]>]]>
```

5.6.1.1.2 <md-compare> using MD-CLI format

Example

```
<rpc message-id="121" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <action xmlns="urn:ietf:params:xml:ns:yang:1">
        <global-operations xmlns="urn:nokia.com:sros:ns:yang:sr:oper-global">
            <md-compare>
                <format>md-cli</format>
            </md-compare>
        </global-operations>
    </action>
</rpc>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="121" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1" xmlns:nokiaoper="urn:nokia.com:sros:ns:yang:sr:oper-
global">
    <nokiaoper:operation-id>12</nokiaoper:operation-id>
    <nokiaoper:start-time>2022-06-20T22:09:24.6Z</nokiaoper:start-time>
    <nokiaoper:results>
        <nokiaoper:md-compare-output>
    configure {
        log {
+           log-id "50" {
+               admin-state enable
+               description "Example"
+               source {
+                   main true
+                   change true
+               }
+               destination {
+                   cli {
+                   }
+               }
+           }
        }
    }
        </nokiaoper:md-compare-output>
    </nokiaoper:results>
```

```
    <nokiaoper:status>completed</nokiaoper:status>
    <nokiaoper:end-time>2022-06-20T22:09:24.6Z</nokiaoper:end-time>
</rpc-reply>
]]>]]>
```

5.6.1.1.3 <md-compare> using XML format and an explicit path

The following example shows the <md-compare> operation run from the **configure log log-id 50 source** path specifying explicitly the XML format.

Example

```
<rpc message-id="121" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1">
    <action xmlns="urn:ietf:params:xml:ns:yang:1">
        <global-operations xmlns="urn:nokia.com:sros:ns:yang:sr:oper-global">
            <md-compare>
                <format>xml</format>
                <path>
                    <subtree-path>
                        <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf">
                            <log>
                                <log-id>
                                    <name>50</name>
                                    <source/>
                                </log-id>
                            </log>
                        </configure>
                    </subtree-path>
                </path>
            </md-compare>
        </global-operations>
    </action>
</rpc>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="121" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:yang=
"urn:ietf:params:xml:ns:yang:1" xmlns:nokiaoper="urn:nokia.com:sros:ns:yang:sr:oper-
global">
    <nokiaoper:operation-id>14</nokiaoper:operation-id>
    <nokiaoper:start-time>2022-06-20T22:12:18.6Z</nokiaoper:start-time>
    <nokiaoper:results>
        <nokiaoper:md-compare-output>
            <configure xmlns="urn:nokia.com:sros:ns:yang:sr:conf" xmlns:nokia-attr=
"urn:nokia.com:sros:ns:yang:sr:attributes">
                <log>
                    <log-id>
                        <name>50</name>
                        <source>
                            <main>true</main>
                            <change>true</change>
                        </source>
                    </log-id>
                </log>
            </configure>
        </nokiaoper:md-compare-output>
    </nokiaoper:results>
    <nokiaoper:status>completed</nokiaoper:status>
    <nokiaoper:end-time>2022-06-20T22:12:18.6Z</nokiaoper:end-time>
</rpc-reply>
```

```
]]>]]>
```

5.6.1.2 <pyexec> YANG-modeled operation

The pySROS libraries enable Python 3 applications that connect to the SR OS node to perform operations, obtain state data, or make configuration changes. These applications run on the originating system's Python interpreter.

Use the MD-CLI **pyexec** command to execute a Python 3 application with the pySROS libraries using the MicroPython interpreter on the SR OS node.

The **<pyexec>** YANG-modeled operation allows NETCONF clients (both management systems and code) to remotely trigger the execution of a Python application to run on the MicroPython interpreter of the SR OS node.

The **pyexec** MD-CLI command and the **<pyexec>** YANG-modeled operation both accept up to ten command line arguments.

The following table summarizes the available input parameters.

Table 31: <pyexec> input parameters

Input parameter	Description
url	The URL that the Python application is located at. If no specific URL is provided in the input, the system default of cf3:\ is prepended to the URL value. Mandatory.
argument-01	The first command line argument to the Python application. Optional.
argument-02	The second command line argument to the Python application. Optional.
argument-03	The third command line argument to the Python application. Optional.
argument-04	The fourth command line argument to the Python application. Optional.
argument-05	The fifth command line argument to the Python application. Optional.
argument-06	The sixth command line argument to the Python application.

Input parameter	Description
	Optional.
argument-07	The seventh command line argument to the Python application. Optional.
argument-08	The eighth command line argument to the Python application. Optional.
argument-09	The ninth command line argument to the Python application. Optional.
argument-10	The tenth command line argument to the Python application. Optional.

Note: The NETCONF server errors if an argument is omitted, for example, argument-01 and argument-03 are provided but argument-02 is not.

Consider the following simple Python application saved as cf3:\example.py on the SR OS device.

Example: Python application saved on an SR OS device

```
from pysros.management import connect
import sys

def main():
    try:
        connection_object = connect()
    except Exception as error:
        raise SystemError(error) from error
    print("pySROS application running on host:", connection_object.running.get('/nokia-
conf:configure/system/name'))
    print("Arguments:", sys.argv[1::])

main()
```

When executed on the SR OS node using the **pyexec** MD-CLI command, the system displays the following output.

Output example

```
[/]
A:admin@node-2# pyexec example.py ex1 ex2 ex3 ex4 ex5 ex6 ex7 ex8 ex9 ex10
pySROS application running on host: sros-router1
Arguments: ['ex1', 'ex2', 'ex3', 'ex4', 'ex5', 'ex6', 'ex7', 'ex8', 'ex9', 'ex10']
```

On a remote NETCONF client, sending the **<pyexec>** YANG-modeled operation executes the example.py Python application stored on the node using the node's MicroPython interpreter.

Output example

```
<rpc message-id="example" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<action xmlns="urn:ietf:params:xml:ns:yang:1">
  <global-operations xmlns="urn:nokia.com:sros:ns:yang:sr:oper-global">
    <pyexec>
      <url>example.py</url>
      <argument-01>test1</argument-01>
      <argument-02>test2</argument-02>
      <argument-03>test3</argument-03>
      <argument-04>test4</argument-04>
      <argument-05>test5</argument-05>
      <argument-06>test6</argument-06>
      <argument-07>test7</argument-07>
      <argument-08>test8</argument-08>
      <argument-09>test9</argument-09>
      <argument-10>test10</argument-10>
    </pyexec>
  </global-operations>
</action>
</rpc>
]]>]]>
```

The system returns the following response. The bolded section is the same output as that shown on the MD-CLI output with the arguments provided in the NETCONF request entered and returned to the **pyexec** output. This output is encapsulated in the <pyexec-output-block> field of the modeled NETCONF response.

Output example

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="example"
 xmlns:nokiaoper="urn:nokia.com:sros:ns:yang:sr:oper-global">
    <nokiaoper:operation-id>20</nokiaoper:operation-id>
    <nokiaoper:start-time>2023-08-23T19:50:43.7Z</nokiaoper:start-time>
    <nokiaoper:results>
        <nokiaoper:pyexec-output-block>pySROS application running on host: sros-router1
Arguments: ['test1', 'test2', 'test3', 'test4', 'test5', 'test6', 'test7', 'test8',
 'test9', 'test10']
</nokiaoper:pyexec-output-block>
    </nokiaoper:results>
    <nokiaoper:status>completed</nokiaoper:status>
    <nokiaoper:end-time>2023-08-23T19:50:44.0Z</nokiaoper:end-time>
</rpc-reply>
]]>]]>
```

5.6.2 NETCONF operations using the md-cli-raw-command request

In addition to individually YANG-modelled operations described in Individually YANG-modeled operations, SR OS also supports a wide set of operations over NETCONF with the md-cli-raw-command request.

Nokia recommends using an individually modeled YANG operation if it is available for a specific operation. For operations that are not individually YANG-modelled, the md-cli-raw-command can be used.

The command input string accepts a command in the exact format as it would be entered in the MD-CLI.

Example

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
```

```
    <global-operations xmlns="urn:nokia.com:sros:ns:yang:sr:oper-global">
      <md-cli-raw-command>
        <md-cli-input-line>clear router Base interface system statistics</md-cli-
input-line>
      </md-cli-raw-command>
    </global-operations>
  </action>
</rpc>
```

Other examples of commands that can be used as input strings include:

- file list cf1:
- show system information
- tools dump resource-usage
- tools perform cflowd manual-export
- //debug router ldp interface foo

The SR OS NETCONF server workflow to process the md-cli-raw-command request is the following.

1. Open a new temporary MD-CLI session (with the same username as the NETCONF session).
2. Pass the input command to the MD-CLI engine.
3. Return the MD-CLI output to the NETCONF client as an unstructured block of text in the <rpc-reply> message.

The MD-CLI context for the operation is the root and the MD-CLI executes the command in operational mode, which is similar to a user newly logged into an MD-CLI session.

Interactive commands, or commands that prompt for input, are not supported and result in an error. For example, using "admin reboot" as an input string fails. Using "admin reboot now" is accepted. Other examples of interactive commands that are not supported include:

- enable
- password
- ssh
- telnet

The **cli-engine** command controls the engines allowed to process the input. Because the starting context for any md-cli-raw-command is the root of the MD-CLI engine, any configuration value that does not allow **md-cli** access causes md-cli-raw-command requests to fail. Access to classic CLI commands through the md-cli-raw-command (for example, commands starting with "//") requires access to both **md-cli** and **classic-cli**. Changes to the **cli-engine** configuration only take effect on raw-md-cli-command in NETCONF sessions that are started after the **cli-engine** configuration was changed.

Only a single operation is supported as the input to the md-cli-raw-command request. Multiple operations require multiple NETCONF RPCs.

The md-cli-raw-command request is not intended as a mechanism to read structured state data or to manage basic configuration. The YANG-modeled configuration and state data is managed and accessed using standard NETCONF operations, such as <edit-config>, <get-config>, and <get>.

The following MD-CLI commands and similar commands are not supported as input strings for md-cli-raw-command:

- admin show configuration
- bof

- configure
- debug
- edit-config
- environment
- exec
- file edit
- history
- info
- li
- logout
- pyexec (when invoked from the pySROS Python 3 libraries)
- //admin display-config
- //admin compare
- //admin rollback
- //admin view
- //bof
- //candidate
- //configure
- //environment
- //exec
- //file vi
- //history
- //logout

Unstructured state information can be retrieved using md-cli-raw-command, for example, with **show** or **tools** dump commands as the input string. The output returned, however, is an unstructured block of text. Structured state information can be retrieved using the standard NETCONF <get> operation.

6 Python

Python is an established programming language that is widely used in systems administration. It has become one of the most prominent languages for network automation.

Python is popular because of its extensive libraries that software developers can import and enhance, and because it offers programming simplicity without diluting the flexibility and scalability offered by other languages.

Note: This chapter assumes prior knowledge and experience with the Python programming language.

6.1 Python in SR OS

SR OS uses version 2 and version 3 of the Python language.

Python 2 and Python 3 are available for ESM applications. Syslog supports Python 2 only. For more information, see Customizing Syslog messages using Python.

Python 3 is available for Python applications run in the MD-CLI using the **pyexec** or **alias** commands, and for applications executed remotely. Python 2 is not supported for **pyexec**, **alias**, CRON, or EHS in the MD-CLI.

The remainder of this section references the SR OS Python 3 functionality only.

6.1.1 Prerequisites

The SR OS node must be in model-driven management interface configuration mode to run Python 3 applications.

To execute Python 3 applications on a remote device that connects to an SR OS device, the following prerequisites must be met:

- On the SR OS node, the NETCONF protocol must be enabled.
- The user must have the required permissions to use the NETCONF protocol.
- On the remote device (that is, the device that executes the application), a Python 3 interpreter of version 3.10 or newer is required.

6.1.2 YANG model support

The Python 3 and pySROS features support Nokia YANG modules (combined and split) and OpenConfig YANG modules. If the YANG modules are to be obtained from the SR OS device, the YANG module files must be available at the specified schema path. See NETCONF monitoring for more information.

6.2 Python 3 interpreter

The Python interpreter is software that compiles and runs applications that developers create. It also handles importing additional libraries and all syntax for the language itself.

SR OS provides the MicroPython interpreter on the SR OS node. This interpreter implements Python 3.4 and is designed to operate using a small memory footprint, which allows SR OS to launch multiple interpreters in parallel.

The Python 3 interpreter is called by the SR OS function that requires it; for example, the MD-CLI. The interpreter is not called directly by the operator.

6.3 Building an application

Customized applications can be written to run on an SR OS node or remotely.

Developers can create applications using any development environment, including a Python 3-based Integrated Development Environment (IDE) or a text editor.

Python 3 applications written for SR OS are designed to be portable between systems (SR OS and operator devices), provided the libraries used and language features are available on both systems.

6.3.1 Available libraries

The following table describes the available preinstalled Python libraries that SR OS provides.

Note: Libraries prefixed by the letter "u" are MicroPython-specific libraries. Importing to a script using the base name is typically possible; for example, **import json** imports ujson when running on an SR OS device.

Table 32: Available libraries

Library	Description
binutils	Collection of binary utilities
datetime	Manipulation of various formats for date, time, and time zone. The **strptime** function is not supported.
ubinascii	Bidirectional translation between binary data and various ASCII encodings
ujson	Conversion between Python objects and JSON data format
pysros	Model-driven management API for SR OS (Python for the Service Router Operating System – pySROS)
ucollections	Advanced collection and container types to hold or accumulate various objects

Library	Description
ure	Simple regular expressions
sys	System-specific parameters and functions
uhashlib	Binary data hashing algorithms
ustruct	Pack and unpack primitive data types
ipaddress	IPv4 and IPv6 address manipulation
uio	Input and output streams
utime	Obtain current date and time, measuring intervals, and delays. This function has been updated to provide support for SR OS and is documented in the API documentation.

6.3.2 The pySROS API

Python for the Service Router Operating System (pySROS) libraries provide the programmatic interface to the SR OS model-driven management framework.

The libraries provide a set of methods for developers to obtain, manipulate, and deliver data to and from an SR OS node.

Note: Nokia recommends reviewing the API documentation provided with the pySROS libraries prior to developing any Python applications.

6.3.2.1 Installing the pySROS libraries

The pySROS libraries are delivered and preinstalled on SR OS nodes. This libraries can be installed on another device, such as a PC or UNIX workstation.

Note: A Python 3 interpreter must be already installed on the device.

Use one of the following methods to install the pySROS libraries:

1. from the Python Package Index (PyPI) using the pip tool (preferred method)

   To do this, execute the **pip install pysros** command on the device.

2. from source code delivered from github.com

   To do this, clone the source code repository to your device by executing the **git clone https://github.com/nokia/pysros** command.

   The source code can then be compiled and installed by executing **python3 setup.py install**.

3. from source code delivered from the Nokia online support portal

   To do this, obtain the source code from the Nokia online support portal. This is the same location where the SR OS software resides.

   The source code can then be compiled and installed by executing **python3 setup.py install**.

6.3.2.2 Importing into applications

The pySROS libraries must be imported to use the provided API in developed applications.

The libraries may be imported using the **import pysros** command, or only specific elements of the API may be imported; for example, from **pysros.management import connect**.

6.3.2.3 API documentation

This chapter provides a high-level overview about how to connect to the model-driven interfaces of an SR OS device and manipulate simple data structures. Detailed technical documentation for the pySROS API is provided within the source code bundle in the `docs` directory.

The API documentation is provided as source code and should be built as needed. To compile the API documentation, run the following commands from the `docs` directory:

- **pip install -r requirements.txt**
- **make html**

The API documentation is compiled into the `build/html` folder in HTML format.

Note: Nokia recommends reviewing the API documentation provided with the pySROS libraries before developing any Python applications.

6.3.2.4 Connecting to a model-driven SR OS device

To enable access to an SR OS device, the user must create a Connection object, which is a Python object that references the application's connection to a specific SR OS node.

The underlying connection transport protocol is NETCONF.

The connect() API method is provided to create this Connection object.

The following example shows the creation of a connection named `connection_object` to a device with the IP address 192.168.168.1.

```
from pysros.management import connect
connection_object = connect(host='192.168.168.1', username='admin', password='admin')
```

The following table describes the arguments available for the connect() API method.

Note: All arguments are ignored when the Python application is executed on an SR OS device.

Table 33: The connect() API method arguments

Argument	Type	Mandatory	Description
host	string	Yes	IP address, hostname, or FQDN of the SR OS device to connect to
username	string	Yes	Username to connect to the SR OS device

Argument	Type	Mandatory	Description
password	string	Yes	Password to connect to the SR OS device If no password is provided, SSH key-based authentication is attempted.
port	integer	No	TCP port on the SR OS node that the connect attempt is made to Default: 830
yang_directory	string	No	Directory to read YANG module files from Default: None
rebuild	Boolean	No	Forces the schema cache to be rebuilt upon connect Default: False
timeout	integer	No	Maximum time for connection, obtaining YANG modules, and schema generation Default: 300 seconds
hostkey_verify	Boolean	No	Disables SSH hostkey checking. Nokia recommends to not use this in a live deployment.
use_existing_ candidate	Boolean	No	Use the existing sessions candidate (supported only on SR OS devices when using a pre- or post-commit Python script)

When a Python application makes a connection to an SR OS device using the connect() method, the application performs the following actions.

- The device is queried to identify a set of modules required to manage the device. This is performed by identifying the module-set-id (yang-library:1.0) or content-id (yang-library:1.1) from the devices advertised capabilities.
- The set of modules are queried using /modules-state from the IETF YANG library RFC 8525.
- The local schema cache is checked to determine whether a model-driven management schema is already generated for the same set of YANG modules. If no identical local schema cache exists, or if the rebuild argument is set to True, the following occurs.
  - If the yang_directory argument is not provided to the connect() API method, each YANG module listed by the SR OS node is downloaded directly from the node.
  - If the yang_directory argument is provided to the connect() API method, the YANG module files from this directory that correspond to the list of required modules obtained from the router are used.
- The YANG modules are compiled into a schema cache for use by the developer and the pySR0S libraries.

Note: The connect() API method is slower to complete for the first compilation of a specific set of YANG modules. Subsequent connections for that set of YANG modules are significantly faster. A locally cached set of YANG modules is not specific to an individual router.

The local model-driven schema cache is stored in the `.pysros` directory within the developer's local home directory. If this directory is deleted or modified, the cache is rebuilt on the next connect() API call.

6.3.2.5 Obtaining modeled data

Use the get() API method to obtain modeled data from an SR OS device.

The get() API method accepts as inputs a JSON instance identifier path and a datastore. The method returns a Python data structure, which uses the pySROS format to provide modeled data to the developer. See The pySROS data structure for more information.

A JSON instance path is obtained from an SR OS node using the **pwc json-instance-path** command. See the *7705 SAR Gen 2 MD-CLI User Guide* for more information.

Example: Obtaining data from an SR OS node Connection object

```
my_data = connection_object.running.get("/nokia-state:state/system/oper-name")
```

Note: State information is obtained from the running the datastore when using the pySROS libraries.

6.3.2.6 Performing operations with pySROS

In addition to device configuration and state information collection, operators can perform the following types of node operations:

- a YANG-modeled operation (defined as an 'action' in a YANG model), which uses structured data for both input and output
- an MD-CLI command, which provides semi-structured input and unstructured output

Nokia recommends using the action() API method when performing operations with SR OS. Some cases may require using an unmodeled MD-CLI command, which can be performed using the cli() API method and manipulating the resulting string output.

6.3.2.6.1 YANG-modeled operations with pySROS

The action() API method allows developers to pass YANG-modeled, structured-data inputs to an operation to be executed on the router and to receive YANG-modeled, structured-data output.

The following example shows a ping operation from the pySROS API.

```
>>> connection_object.action('/nokia-oper-global:global-operations/ping', {'destination':
 '192.168.122.1'})

Action({'operation-id': Leaf(13), 'start-time': Leaf('2023-10-04T19:00:20.1Z'), 'results':
 Container({'test-parameters': Container({'destination': Leaf('192.168.122.1'), 'bypass-
routing': Leaf(False), 'router-instance': Leaf('Base'), 'srv6-policy': Leaf(False), 'candidate-
path': Leaf(False), 'preference': Leaf(100), 'count': Leaf(5), 'output-format': Leaf('detail'),
 'do-not-fragment': Leaf(False), 'fc': Leaf('nc'), 'interval': Leaf('1'), 'pattern':
```

```
 Leaf('sequential'), 'size': Leaf(56), 'timeout': Leaf(5), 'tos': Leaf(0), 'ttl': Leaf(64)}),
 'probe': {1: Container({'probe-index': Leaf(1), 'status': Leaf('no-route-to-dest')}), 2:
 Container({'probe-index': Leaf(2), 'status': Leaf('no-route-to-dest')}), 3: Container({'probe-
index': Leaf(3), 'status': Leaf('no-route-to-dest')}), 4: Container({'probe-index': Leaf(4),
 'status': Leaf('no-route-to-dest')}), 5: Container({'probe-index': Leaf(5), 'status':
 Leaf('no-route-to-dest')})}, 'summary': Container({'statistics': Container({'packets':
 Container({'sent': Leaf(5), 'received': Leaf(0), 'loss': Leaf('100.0')}), 'round-trip-time':
 Container({'minimum': Leaf(0), 'average': Leaf(0), 'maximum': Leaf(0)})})})}), 'status':
 Leaf('completed'), 'end-time': Leaf('2023-10-04T19:00:24.3Z')})
```

6.3.2.6.2 MD-CLI command execution with pySROS

The cli() API method allows developers to pass a single line string containing a single MD-CLI command (including output modifiers, if required) to an existing SR OS connection object. SR OS executes this command immediately and returns the output as a string. See NETCONF operations using the md-cli-raw-command request for more information about supported commands.

The following example executes an MD-CLI command from the pySROS API.

```
output_string = connection_object.cli('show system time')
print(output_string)
===============================================================================
Date & Time
===============================================================================
Current Date & Time : 2021/09/15 20:27:27    DST Active            : no
Current Zone        : UTC                    Offset from UTC       : 00:00
-------------------------------------------------------------------------------
Non-DST Zone        : UTC                    Offset from UTC       : 00:00
Zone type           : standard
-------------------------------------------------------------------------------
No DST zone configured.
-------------------------------------------------------------------------------
Prefer Local Time   : NO
===============================================================================
```

6.3.2.7 Editing configuration

Python data structures, in pySROS format, and specific values within the data structures can be configured on an SR OS device.

The set() API method is provided to configure a node. The set() API method aggregates several model-driven transactional operations into a single method:

- configuration of the node (performed in private candidate configuration mode)
- update, validate, and commit

The set() API method accepts as input a JSON instance path, a datastore, and a payload. The payload may be a single value, a key and value pair, a crafted Python data structure following the pySROS data-structure format, or a pySROS data structure object received from a get() API method call.

Example: Setting the system name to "hostname"

```
connection_object.candidate.set("/nokia-conf:configure/system/name", "hostname")
connection_object.candidate.set("/nokia-conf:configure/system", {'name':'hostname'})
connection_object.candidate.set("/nokia-
conf:configure", {'system': {'name': 'hostname'}})
```

6.3.3 The pySROS data structure

The key to efficient Python programs for network management is the ability to manipulate data in a consumable format.

The pySROS libraries ensure that the YANG modeled data from SR OS is supplied in native Python data structures that are simple to understand and handle in code.

When data is obtained from an SR OS node, it is returned in a specific data structure format, a pySROS data structure.

The following table shows the pySROS conversion from model-driven YANG-based data structures to native Python data structures.

Table 34: Model-driven to Python data structure conversion rules

YANG structure	Python 3 structure
Container	Dict
Leaf	Value (Type derived as shown in Table 35: YANG types)
Leaf-list	List
List	Dict
User-ordered List	OrderedDict keyed on the YANG list key value

The pySROS libraries provide Python class wrappers around each node type to assist with data manipulation. This information can be used with some of the features built into the pySROS API, such as pretty-printing.

Containers are wrapped in a Container() class.

Leafs are wrapped in a Leaf() class.

Example: Command usage to obtain the value of a leaf

The following example shows how to obtain the value of the leaf that is wrapped in a Leaf() class by calling the .data function on the leaf.

```
from pysros.wrappers import Leaf
obj = Leaf('example')
print(obj.data)
```

Leaf-lists are wrapped in a LeafList() class.

All pySROS class wrappers for YANG nodes are provided by **pysros.wrappers**.

YANG schema metadata can be obtained on a specific data object by calling the schema function. The YANG metadata that is available within the pySROS data structure (data is the specific pySROS formatted data object) is the data.schema.module. This is the YANG module name the element comes from. If this element in the YANG schema is imported or included from another YANG modules, the root YANG module is displayed. If the YANG modules where the element is found is an augment to another module, the augment module is shown.

YANG types are also converted into native Python types. The following table describes the rules for this conversion.

Table 35: YANG types

Base YANG type	Python 3 type
binary	String
bits	String
boolean	Boolean
decimal64	String
empty	[13]
enumeration	String
identityref	String
int8	Integer
int16	Integer
int32	Integer
int64	Integer
leafref	(not applicable [14])
string	String
uint8	Integer
uint16	Integer
uint32	Integer
uint64	Integer
union	String [15]

13 The specific type is provided by the pySROS libraries.

14 A leafref takes the YANG native type of the leaf it is referencing. This type is converted to Python according to the table.

15 A union YANG type may be a union of different YANG types, for example, a union of a string and a Boolean. As it is not possible to identify the intention at the time of obtaining the data, automatic type selection is not performed. Every union is treated as a string, allowing the developer to cast the element into a specified type.

6.4 Provisioning and precompiling Python applications

Python applications can be configured and executed in the MD-CLI. This allows administrators to provide specific Python applications to users.

Python applications can be referenced in configuration statements, such as MD-CLI aliases.

Python applications are configured in the **configure python** context.

Example: Python application configuration

```
(ex)[/configure python]
A:admin@node-2# info
    python-script "my_example" {
        admin-state enable
        urls ["cf3:\example.py" "ftp://user:password@192.168.168.254/example.py"]
        version python3
    }
```

After a Python application is configured and committed, the SR OS node attempts to load the files specified in the `urls` field in the order they are listed. If the first URL location can be reached and a file containing valid Python code is found, the system stores this source code ready for use.

If the first URL cannot be reached or the file does not contain valid source code, the system moves to the next URL in the list. If no URL locations contain valid Python source code, the **oper-state** leaf for the configured application is set to down in the following context.

```
state python python-script application_name
```

6.4.1 Refreshing configured Python scripts

When defined in the SR OS configuration, Python scripts are stored in memory on the SR OS node. The URLs defined in the configuration are not rechecked, unless it is specifically requested by an operator.

If a new version of the script is placed in the URL reference location, whether locally on the SR OS nodes storage card or remotely, the following command must be run to use the new version.

```
tools perform python-script reload application_name
```

6.5 Execution pathways

This section describes the methods that are provided to execute Python applications.

6.5.1 Executing a Python application remotely

The pySROS libraries provide the ability to interface with one or more SR OS devices from a device that has a Python interpreter installed. This execution pathway is not specific to SR OS (beyond the use of the pySROS libraries) and is not described further in this guide.

6.5.2 Executing a Python application from the command line

The MD-CLI provides the ability to execute a Python application directly from the command line.

The **pyexec** command takes as an option, either the name of a Python application from the following SR OS configuration or the URL to the location (local/remote) of a Python application.

```
configure python python-script application_name
```

For example, if the **pyexec** *application_name* command is executed, the SR OS performs the following steps:

1. The SR OS device searches for a configuration element **configure python python-script** *application_name*. If a configuration element is found, the system attempts to compile and execute this script. A failure ends the execution.
2. The SR OS device treats the *application_name* as a URL. As *application_name* does not have a URL identifier (such as `cf3:\` or `ftp://`) attached to it, SR OS prepends the present working directory of the system (which is `cf3:\` by default on most systems).
3. SR OS attempts to read from the derived URL; for example, `cf3:\application_name`. If this URL is successfully located, SR OS compiles and runs the application. A failure ends the execution.

   Each time **pyexec** is invoked, a new Python interpreter is spawned.

6.5.2.1 pyexec and input parameters

The **pyexec** command can accept a maximum of ten input arguments provided using the MD-CLI command line. The Python application developer must handle these as standard input (STDIN) arguments.

6.5.2.2 Executing a Python application as an output modifier

Python applications can be entered as an output modifier to the MD-CLI commands.

The following is an example output from an **info** command that is processed by a Python application, which capitalizes the output.

Example

```
(ex)[/]
A:admin@node-2# file show cf3:\capitals.py
File: capitals.py
-------------------------------------------------------------------------------
import sys
def main():
  for line in sys.stdin:
    print(line.upper(), end="")
main()
===============================================================================
(ex)[/]
A:admin@node-2# show version | pyexec capitals.py
TIMOS-B-21.7.B1-9 BOTH/X86_64 NOKIA 7750 SR COPYRIGHT (C) 2000-2021 NOKIA.
ALL RIGHTS RESERVED. ALL USE SUBJECT TO APPLICABLE LICENSE AGREEMENTS.
BUILT ON THU JUN 12 18:35:55 PDT 2021 BY BUILDER IN /BUILDS/C/217B/B1-9/PANOS/MAIN/SROS
```

6.5.2.3 Executing Python automatically before or after commit operations

Python applications can be executed automatically before issuing the **commit** command, after issuing the **commit** command, or both, through the MD-CLI, NETCONF, or gNMI. This functionality is only available for Python applications stored locally on the SR OS compact flash devices and configured under **/configure python python-script**.

Commit scripts are configured in the MD-CLI as in the following example, where the pre-commit script is stored on cf3: and is named pre.py. The post-commit script is also stored on cf3: and is named post.py.

```
/configure python python-script "post-script" admin-state enable
/configure python python-script "post-script" urls ["cf3:/post.py"]
/configure python python-script "post-script" version python3
/configure python python-script "pre-script" admin-state enable
/configure python python-script "pre-script" urls ["cf3:/pre.py"]
/configure python python-script "pre-script" version python3
/configure system management-interface commit-management python-scripts pre-commit-python-
script "pre-script"
/configure system management-interface commit-management python-scripts post-commit-python-
script "post-script"
```

By default, any configured (and operationally up) pre- or post-commit script runs when a commit is triggered from the MD-CLI, NETCONF, or gNMI. The default configuration values are as follows:

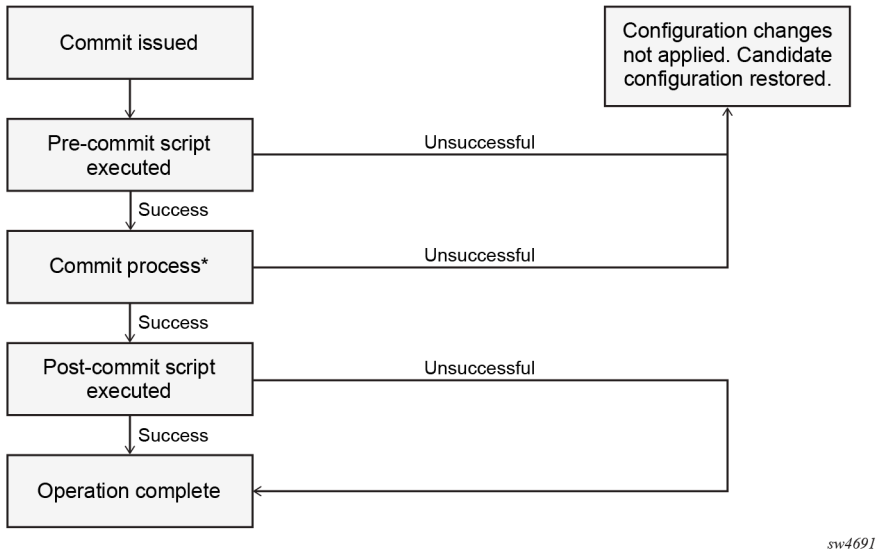
```
/configure system management-interface commit-management python-scripts md-cli-trigger true
/configure system management-interface commit-management python-scripts netconf-trigger true
/configure system management-interface commit-management python-scripts gnmi-trigger true
```

To alter the behavior on a specific interface, set the value to **false**.

Commit scripts are supported for both **commit** and **commit confirmed** operations.

A single commit operation behaves as follows:

Figure 19: Commit operation



The success or failure of the pre-commit script determines whether the commit process is attempted. An example use case for this functionality is to check that a candidate configuration conforms to corporate standards before committing.

The success or failure of the post-commit script does not affect the commit itself, but the result is recorded in the device logs.

The commit process (highlighted with the * in the diagram above) includes the following activities in a **commit** situation:

1. Update from baseline
2. Validate the configuration
3. Copy the candidate configuration to the running configuration
4. Reset the candidate configuration to match the new running configuration ready for further changes
5. Save the configuration (if **auto-save** is enabled)

The commit process (highlighted with the * in the diagram above) includes the following activities in a **commit confirmed** situation:

1. Update from baseline
2. Validate the configuration
3. Copy the candidate configuration to the running configuration
4. Reset the candidate configuration to match the new running configuration ready for further changes
5. Start the commit confirmed timer
6. Receive the accept command for the commit confirmed
7. Save the configuration (if **auto-save** is enabled)

If the commit process (highlighted with the * in the figure above) fails to complete successfully (for example, because of a validation error, the commit confirmed timer expires, or the operator cancels the in-progress commit confirmed timer), the commit does not complete, the running configuration remains unchanged, and the candidate configuration still contains the proposed configuration changes.

If the pre- or post-commit scripts cannot be run because they are administratively or operationally down, they are skipped and the system raises a log message. These log events are suppressed by default.

A commit script can access the existing candidate of the current user's session by providing the **use_existing_candidate=True** argument to the **connect()** pySROS method. A script can analyze (and optionally change in a pre-commit script) the configuration between an operator issuing a commit and the configuration applied to the system. To manipulate the configuration in a pre-commit script, use the pySROS **set()** method with the **commit=False** argument.

The entire process shown above in the diagram is considered a single commit. Therefore, SR OS locks the running and candidate configurations for the duration of this single commit process. This should be considered when deciding whether to deploy long-running commit scripts.

Note: If the user that accepts an ongoing confirmed commit is a different user from the user that initially issued the commit, the post-commit script will run using the accepting user (unless **run-as-user** is configured). Any authorization triggered as a result of the post-commit script will be completed as the accepting user.

Commit scripts are subject to the same configurable timeout and memory consumption options as any other Python application on SR OS.

Commit scripts support the **run-as-user** option if an administrator needs to execute scripts with a different privilege level than the operator issuing the commit.

To disable the pre- and post-commit scripts for the next commit operation, use **/admin system management-interface commit-management python-scripts disable-next-run**. This sets **/state system management-interface commit-management python-scripts next-commit** to **scripts-will-not-execute**. This command prevents pre- and post-commit scripts from executing until the next commit operation completes. Upon completion, the state leaf resets, enabling pre- or post-commit scripts to run for the subsequent commit.

6.5.2.4 Chaining Python applications

The **pyexec** command supports chaining multiple applications on the command line. Python applications called with the **pyexec** command can produce data on the standard output (STDOUT) stream and consume data on the standard input (STDIN) stream. Multiple Python applications can then be chained together.

Note: A maximum of three **pyexec** commands may be chained together.

6.5.2.5 Python applications and the MD-CLI pager

Python applications executed using the **pyexec** command use the built-in MD-CLI pager to display the output one page at a time. This behavior can be disabled per Python application using the **no-more** output modifier, or globally by setting the MD-CLI **environment more** option to **false**, as required.

When the MD-CLI display output is paused by the pager while executing a Python application, pressing **q** closes the pager and resumes executing the Python application. Pressing **Ctrl-c** closes the pager and terminates the Python application.

6.5.3 Executing Python applications with MD-CLI command aliases

Python applications can be executed from a configured alias in MD-CLI. See "Command Aliases" in the *7705 SAR Gen 2 MD-CLI User Guide* for more information about how to configure an alias in the MD-CLI.

Aliases may call a configured (named) Python application or the **pyexec** command.

There are specific AAA considerations for each method. See Authentication, authorization, and accounting for more information.

Each time an alias that references a Python application is executed, a new Python interpreter is spawned.

6.6 Security

SR OS ensures that the routing, forwarding, and management functions of a device are protected and prioritized.

6.6.1 Python authentication and authorization

Python applications are dependent on authentication, authorization, and accounting. See Authentication, authorization, and accounting for more information.

The user associated with the current session is used for authentication, authorization, and accounting of the script, as well as the authorization of the content of the scripts results, and any resulting file access.

If the following command is configured, the specified user is used for all authentication, authorization, accounting, and file access of the script, instead of the current session user.

```
configure python python-script run-as-user
```

The **run-as-user** command is configured on a per python-script basis and takes precedence over any other user configured on a less granular level (for example, CRON or EHS).

Note: If the **run-as-user** command is specified, Nokia recommends paying particular attention to the authorization rules about who is allowed to execute the configured **python-script**.

6.6.1.1 Authentication

To make a connection to the model-driven interface of an SR OS device, user authentication is required.

User authentication is deemed to have already occurred if the Python application is executed directly on the SR OS node.

If the Python application is executed remotely, user credentials must be provided to the connect() API call.

6.6.1.2 Authorization

To successfully use a Python application, the authenticated user must have authorization to both execute the application and perform each of the model-driven operations contained within the Python application.

6.6.2 Other restrictions

To protect the integrity of an SR OS system, a number of facilities are disabled in Python while executing an application on SR OS. These include, but are not limited to the following:

- importing additional libraries outside of the supported list
- creating, reading, writing, or otherwise manipulating network sockets
- garbage collection
- debugging libraries

6.6.3 Memory management

SR OS does not allow a Python application to start if there is insufficient memory available at the point of execution, or if too many Python applications are running concurrently.

When a Python 3 application is spawned, SR OS confirms that the minimum percentage of sufficient free system memory is available. The percentage value for minimum available system memory is configurable within the MD-CLI environment settings.

The Python 3 application then reserves memory incrementally (and automatically) for the application up to the value of the maximum memory configured in the MD-CLI environment settings.

6.6.4 Performance and scale

The ability to execute Python applications conveys no expectation of a specific performance or scale level.

Note: Nokia recommends using smaller data sets, where possible, when executing a Python application on an SR OS device. For larger data sets, Nokia recommends executing Python applications remotely.

7 Event and accounting logs

This chapter provides information about configuring event and accounting logs on the SR OS.

7.1 Logging overview

The two primary types of logging supported in the SR OS are event logging and accounting logs.

Event logging controls the generation, dissemination, and recording of system events for monitoring status and troubleshooting faults within the system. The SR OS groups events into four major categories or event sources:

- **security events**

  Events that pertain to attempts to breach system security.
- **change events**

  Events that pertain to the configuration and operation of the node.
- **main events**

  Events that pertain to applications that are not assigned to other event categories or sources.
- **debug events**

  Events that pertain to trace or other debugging information.

Events within the SR OS have the following characteristics:

- timestamp in UTC or local time
- generating application
- unique event ID within the application
- router name (also called a vrtr-name) identifying the associated routing context (for example, Base or vprn1000)
- subject identifying the affected object for the event (for example, interface name or port identifier)
- short text description

Event control assigns the severity for each application event and whether the event will be generated or suppressed. The severity numbers and severity names supported in the SR OS conform to ITU standards M.3100 X.733 and X.21 and are listed in the following table.

Table 36: Event severity levels

Severity number	Severity name
1	cleared
2	indeterminate (info)
3	critical

Severity number	Severity name
4	major
5	minor
6	warning

Events that are suppressed by event control will not generate any event log entries. Event control maintains a count of the number of events generated (logged) and dropped (suppressed) for each application event. The severity of an application event can be configured in event control.

An event log within the SR OS associates event sources with logging destinations. Logging destinations include the following:

- console session
- Telnet or SSH session
- memory logs
- file destinations
- SNMP trap groups
- syslog destinations

A log filter policy can be associated with the event log to control which events are logged in the event log based on combinations of application, severity, event ID range, router name (vrtr-name), and the subject of the event.

The SR OS accounting logs collect comprehensive accounting statistics to support a variety of billing models. The routers collect accounting data on services and network ports on a per-service class basis. In addition to gathering information critical for service billing, accounting records can be analyzed to provide insight about customer service trends for potential service revenue opportunities. Accounting statistics on network ports can be used to track link utilization and network traffic pattern trends. This information is valuable for traffic engineering and capacity planning within the network core.

Accounting statistics are collected according to the options defined within the context of an accounting policy. Accounting policies are applied to access objects (such as access ports and SAPs) or network objects (such as network ports and network IP interface). Accounting statistics are collected by counters for individual service meters defined on the customer SAP or by the counters within forwarding class (FC) queues defined on the network ports.

The type of record defined within the accounting policy determines where a policy is applied, what statistics are collected, and time interval at which to collect statistics.

The supported destination for an accounting log is a compact flash system device. Accounting data is stored within a standard directory structure on the device in compressed XML format. On platforms that support multiple storage devices, Nokia recommends that accounting logs be configured on the cf1: or cf2: devices only. Accounting log files are not recommended on the cf3: device if other devices are available (Nokia recommends that cf3: be used primarily for software images and configuration related files).

7.2 Log destinations

Both event logs and accounting logs use a common mechanism for referencing a log destination.

The SR OS supports the following log destinations:

- console
- session
- CLI logs
- memory logs
- log files
- SNMP trap group
- syslog
- NETCONF

Only a single log destination can be associated with an event log or accounting log. An event log can be associated with multiple event sources, but it can only have a single log destination. An accounting log can only have a file destination.

7.2.1 Console

Sending events to a console destination means the message is sent to the system console The console device can be used as an event log destination.

7.2.2 Session

A session destination is a temporary log destination which directs entries to the active Telnet or SSH session for the duration of the session. When the session is terminated, for example, when the user logs out, the "to session" configuration is removed. Event logs configured with a session destination are stored in the configuration file but the "to session" part is not stored. Event logs can direct log entries to the session destination.

7.2.3 CLI logs

A CLI log is a log that outputs log events to a CLI session. The events are sent to the CLI session for the duration of that CLI session (or until an **unsubscribe-from** command is issued).

Use the following command to subscribe to a CLI log from within a CLI session.

```
tools perform log subscribe-to log-id
```

7.2.4 Memory logs

A memory log is a circular buffer. When the log is full, the oldest entry in the log is replaced with the new entry. When a memory log is created, the specific number of entries it can hold can be specified, otherwise it assumes a default size. An event log can send entries to a memory log destination.

7.2.5 Log and accounting files

Log files can be used by both event logs and accounting logs and are stored on the compact flash devices in the file system.

A log file policy is identified using a numerical ID in classic interfaces and a string name in MD interfaces, but a log file policy is generally associated with a number of individual files in the file system. A log file policy is configured with a rollover parameter, expressed in minutes, which represents the period of time an individual log file is written to before a new file is created for the relevant log file policy. The rollover time is checked only when an update to the log is performed. Therefore, complying to this rule is subject to the incoming rate of the data being logged. For example, if the rate is very low, the actual rollover time may be longer than the configured value.

The retention time for a log file policy specifies the period of time an individual log file is retained on the system based on the creation date and time of the file. The system continuously checks for log files with expired retention periods once every hour and deletes as many files as possible during a 10-second interval.

When a log file policy is created, only the compact flash device for the log files is specified. Log files are created in specific subdirectories with standardized names depending on the type of information stored in the log file.

Event log files are always created in the `\log` directory on the specified compact flash device. The naming convention for event log files is:

`log eeff-timestamp`

where

- *ee* is the event log ID
- *ff* is the log file destination ID
- *timestamp* is the timestamp when the file is created in the form of:

  yyyymmdd-hhmmss

  where

  – *yyyy* is the four-digit year (for example, 2019)

  – *mm* is the two-digit number representing the month (for example, 12 for December)

  – *dd* is the two-digit number representing the day of the month (for example, 03 for the 3rd of the month)

  – *hh* is the two-digit hour in a 24-hour clock (for example, 04 for 4 a.m.)

  – *mm* is the two-digit minute (for example, 30 for 30 minutes past the hour)

  – *ss* is the two-digit second (for example, 14 for 14 seconds)

Accounting log files are created in the `\act-collect` directory on a compact flash device (specifically `cf1` or `cf2`). The naming convention for accounting log files is nearly the same as for log files except the prefix `act` is used instead of the prefix `log`. The naming convention for accounting logs is:

`act aaff-timestamp.xml.g`

where

- *aa* is the accounting policy ID
- *ff* is the log file destination ID

- *timestamp* is the timestamp when the file is created in the form of:

  yyyymmdd-hhmmss

  where

  – *yyyy* is the four-digit year (for example, 2019)

  – *mm*

    is the two-digit number representing the month (for example, 12 for December)

  – *dd* is the two-digit number representing the day of the month (for example, 03 for the 3rd of the month)

  – *hh* is the two-digit hour in a 24-hour clock (for example, 04 for 4 a.m.)

  – *mm* is the two-digit minute (for example, 30 for 30 minutes past the hour)

  – *ss* is the two-digit second (for example, 14 for 14 seconds)

Accounting logs are XML files created in a compressed format and have a .gz extension.

Active accounting logs are written to the `\act-collect` directory. When an accounting log is rolled over, the active file is closed and archived in the `\act` directory before a new active accounting log file is created in `\act-collect`.

When creating a new event log file on a compact flash disk card, the system checks the amount of free disk space and that amount must be greater than or equal to the lesser of 5.2 MB or 10% of the compact flash disk capacity.

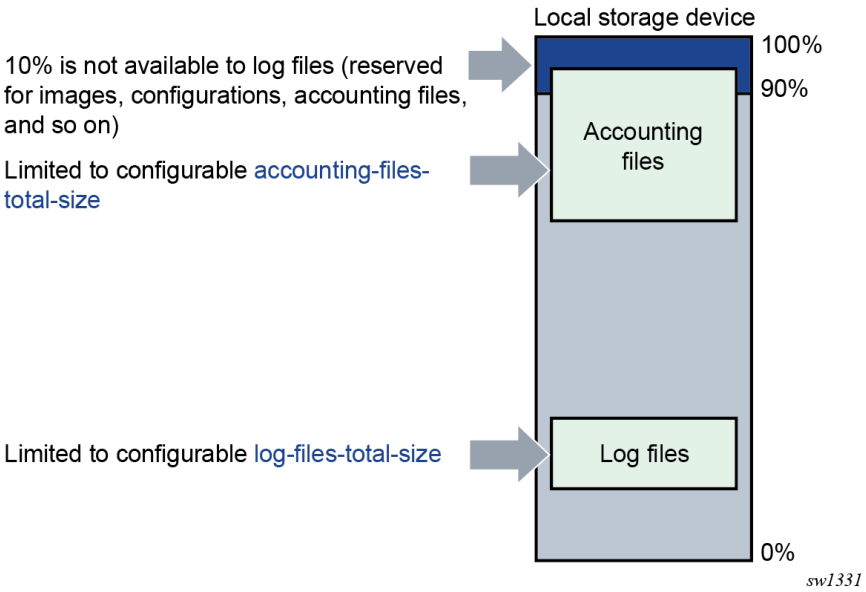
In addition to the 10% free space limit for event log files described in the preceding paragraph, configurable limits for the total size of all system-generated log files and all accounting files on each storage device are available using the following commands.

```
configure log file-storage-control accounting-files-total-size
configure log file-storage-control log-files-total-size
```

The space on each storage device (cf1, cf2, and so on) is independently limited to the same configured value.

The following figure illustrates the file space limits.

Figure 20: Accounting and log file storage limits



The system calculates the total size of all accounting files and log files on each storage device on the active CPM every hour. The storage space used on the standby CPM is not actively managed. If a user manually adds or deletes accounting or log files in the `\act` or `\log` directories, the total size of the files is taken into account during the next hourly calculation cycle. Files added by the system (that is, a new log file after a rollover period ends) or removed by the system (that is, a file that is determined as past the retention time during the hourly checks) are immediately accounted for in the total size.

If the configured limit is reached, the system attempts a cleanup to generate free space, as follows:

1. Completed files beyond their retention time are removed.
2. If the total size of all log files is still above the configured limit for a specific storage device, the oldest completed log files are removed until the total log size is below the limit. Accounting files below their retention time are not removed.

Whether the configurable total size limits are configured or not, log and accounting files never overwrite other types of files, such as images, configurations, persistency, and so on.

7.2.5.1 Log file encryption

The log files saved in local storage can be encrypted using the AES-256-CTR cipher algorithm.

Use the following command to configure the log file encryption key and enable log file encryption. The encryption key is used for all local log files in the system.

```
configure log encryption-key
```

Note:

- When an encrypted log file is opened in a text editor, editing or viewing the file contents is not possible, as the entire file is encrypted.

- The encrypted log files can be decrypted offline using the appropriate OpenSSL command.

```
openssl enc aes-256-ctr -pbkdf2 -d -in <log file encrypted> -out <output log file> -
p -pass pass:<passphrase>
```

7.2.6 SNMP trap group

An event log can be configured to send events to SNMP trap receivers by specifying an SNMP trap group destination.

An SNMP trap group can have multiple trap targets. Each trap target can have different operational values.

A trap destination has the following properties:

- IP address of the trap receiver
- UDP port used to send the SNMP trap
- SNMP version (v1, v2c, or v3) used to format the SNMP notification
- SNMP community name for SNMPv1 and SNMPv2c receivers
- Security name and level for SNMPv3 trap receivers

For SNMP traps that are sent out-of-band through the Management Ethernet port on the SF/CPM, the source IP address of the trap is the IP interface address defined on the Management Ethernet port. For SNMP traps that are sent in-band, the source IP address of the trap is the system IP address of the router.

Each trap target destination of a trap group receives the identical sequence of events as defined by the log ID and the associated sources and log filter applied. For the list of options that can be sent in SNMP notifications, please see the SR OS MIBs (and RFC 3416, section 4.2.6).

7.2.7 Syslog

7.2.7.1 Syslog implementation overview

An event log can be configured to send events to one syslog destination. Syslog destinations have the following properties:

- Syslog server IP address or Fully Qualified Domain Name (FQDN)
- UDP port or TLS profile used to send the syslog message
- Syslog Facility Code (0 to 23) (default 23 - local 7)
- Syslog Severity Threshold (0 to 7); sends events exceeding the configured level

Because syslog uses eight severity levels, whereas the SR OS uses six internal severity levels, the SR OS severity levels are mapped to syslog severities. The following table describes the severity-level mappings.

Table 37: Router to syslog severity-level mappings

SR OS event severity	Syslog severity numerical code	Syslog severity name	Syslog severity definition
—	0	emergency	System is unusable

SR OS event severity	Syslog severity numerical code	Syslog severity name	Syslog severity definition
critical	1	alert	Action must be taken immediately
major	2	critical	Critical conditions
minor	3	error	Error conditions
warning	4	warning	Warning conditions
—	5	notice	Normal but significant condition
cleared or indeterminate	6	info	Informational messages
—	7	debug	Debug-level messages

The general format of an SR OS syslog message is the following, as defined in RFC 3164, *The BSD Syslog Protocol*:

<PRI><HEADER> <MSG>

Note: The "<" and ">" are informational delimiters to make reading and understanding the format easier. They do not appear in the actual syslog message except as part of the PRI.

where:

- <PRI> is a number that is calculated from the message Facility and Severity codes as follows:

  Facility * 8 + Severity

  The calculated PRI value is enclosed in "<" and ">" angle brackets in the transmitted syslog message.
- <HEADER> is composed of the following:

  <TIMESTAMP> <HOSTNAME>

  – <TIMESTAMP> immediately follows the trailing ">" from the PRI part, without a space between. Depending on the configuration of the **configure log syslog timestamp-format** command, the format is either:

  MMM DD HH:MM:SS

  or

  MMM DD HH:MM:SS.sss

  There are always two characters for the day (DD). Single-digit days are preceded with a space character. Either UTC or local time is used, depending on the configuration of the **time-format** command for the event log.

  – <HOSTNAME> follows the <TIMESTAMP> with a space between. It is an IP address by default, or for event logs (but not filter syslogs), can be configured to use other values using the following commands:

```
configure log syslog hostname
configure service vprn log syslog hostname
```

- <MSG> is composed of the following:

<log-prefix>: <seq> <vrtr-name> <application>-<severity>-<Event Name>-<Event ID> [<subject>]: <message>\n

- <log-prefix> is an optional 32 characters of text (default = 'TMNX') as configured using the **log-prefix** command.
- <seq> is the log event sequence number (always preceded by a colon and a space char)
- <vrtr-name> is vprn1, vprn2, … | Base | management | vpls-management
- <subject> may be empty, resulting in []:
- \n is the standard ASCII newline character (0x0A)

Examples (from different nodes)

default log-prefix (TMNX):

```
<188>Jan  2 18:43:23 10.221.38.108 TMNX: 17 Base SYSTEM-WARNING-tmnxStateChange-
2009 [CHASSIS]:  Status of Card 1 changed administrative state: inService,
operational state: outOfService\n
<186>Jan  2 18:43:23 10.221.38.108 TMNX: 18 Base CHASSIS-MAJOR-tmnxEqCardRemoved-
2003 [Card 1]:  Class IO Module : removed\n
```

no log-prefix:

```
<188>Jan 11 18:48:12 10.221.38.108 : 32 Base SYSTEM-WARNING-tmnxStateChange-2009
[CHASSIS]:  Status of Card 1 changed administrative state: inService,
operational state: outOfService\n
<186>Jan 11 18:48:12 10.221.38.108 : 33 Base CHASSIS-MAJOR-tmnxEqCardRemoved-
2003 [Card 1]:  Class IO Module : removed\n
```

log-prefix "test":

```
<186>Jan 11 18:51:22 10.221.38.108 test: 47 Base CHASSIS-MAJOR-tmnxEqCardRemoved-
2003 [Card 1]:  Class IO Module : removed\n
<188>Jan 11 18:51:22 10.221.38.108 test: 48 Base SYSTEM-WARNING-tmnxStateChange-
2009 [CHASSIS]:  Status of Card 1 changed administrative state: inService,
operational state: outOfService\n
```

Syslog IP header source address

The source IP address field of the IP header on syslog message packets depends on a number of factors, including which interface the message is transmitted on and a few configuration commands.

When a syslog packet is transmitted out-of-band (out a CPM Ethernet port in the management router instance), the source IP address contains the address of the management interface as configured in the BOF.

When a syslog packet is transmitted in-band (for example, out a port on an IMM) in the Base router instance, the order of precedence for how the source IP address is populated is the following:

MD-CLI

1. source address

```
configure system security source-address ipv4 syslog
configure system security source-address ipv6 syslog
```

2. system address

```
configure router interface "system" ipv4 primary address
configure router interface "system" ipv6 address
```

3. IP address of the outgoing interface

Classic CLI

1. source address

```
configure system security source-address application syslog
configure system security source-address application6 syslog
```

2. system address

```
configure router interface "system" address
configure router interface "system" ipv6 address
```

3. IP address of the outgoing interface

When a syslog packet is transmitted out a VPRN interface, the source IP address is populated with the IP address of the outgoing interface.

Syslog HOSTNAME

The HOSTNAME field of syslog messages can be populated with an IP address, the system name, or a number of other options.

For event logs, use the following commands if a system name or other string is wanted. This command is ignored for filter syslogs.

```
configure log syslog hostname
configure service vprn log hostname
```

For filter syslogs, and for event logs if the **hostname** command is not configured, SR OS populates the syslog HOSTNAME field with an IP address as follows.

When a syslog packet is transmitted out-of-band (out a CPM Ethernet port in the management router instance), the HOSTNAME field contains the address of the management interface as configured in the BOF.

When a syslog packet is transmitted in-band (for example, out a port on an IMM) in the Base router instance, the order of precedence for how the source IP address is populated is the following:

MD-CLI

1. source address

```
configure system security source-address ipv4 syslog
configure system security source-address ipv6 syslog
```

2. system address

```
configure router interface "system" ipv4 primary address
configure router interface "system" ipv6 address
```

3. lowest loopback address
4. lowest exit address

Classic CLI

1. source address

```
configure system security source-address application syslog
configure system security source-address application6 syslog
```

2. system address

```
configure router interface "system" address
configure router interface "system" ipv6 address
```

3. lowest loopback address
4. lowest exit address

When a syslog packet is transmitted out a VPRN interface, the HOSTNAME is populated with the VPRN loopback address. When more than one loopback exists, the HOSTNAME contains the lowest loopback IP address. If no loopback interface is configured, the HOSTNAME contains the physical exit interface IP address. When no loopback interface is configured and more than one physical exit interface exists, the hostname contains the lowest physical exit interface IP address.

7.2.7.2 Syslog over TLS for log events

Syslog messages containing log events can be optionally sent over TCP and TLS instead of UDP. TLS support for log event syslog messages is based on RFC 5425, which provides security for syslog through the use of encryption and authentication. Use the following command to enable TLS for syslog log events by configuring a TLS profile against the syslog profile.

```
configure log syslog tls-client-profile
```

Syslog over TLS packets are sent with a fixed TCP source port of 6514.

TLS is supported for the following log event syslogs:

- system syslogs (**configure log syslog**), which can send syslog messages as follows:
  - in-band (for example, out a port on an IMM)
  - out-of-band (out a CPM Ethernet port in the management router instance)

  The **configure log route-preference** command configuration determines where the TLS connection is established for the base system syslogs.
- service VPRN syslogs using the following command

```
configure service vprn log syslog
```

TLS is not supported for filter logs.

If the TLS connection to the syslog collector is not established correctly or is broken during operation, the connection is retried each time there is a new log event to send.

7.2.7.3 Syslog over TLS with FQDN

SR OS supports using the FQDN for the syslog server destination. When the FQDN is programmed as the IP address, the FQDN field is used as the destination IP after it is resolved via DNS. The FQDN is also used to verify the SAN field in the server certificate.

The following describes the authentication process when an FQDN is configured as the syslog address:

1. To connect to the server, the router first resolves the FQDN via DNS.
2. After resolving the FQDN, the router uses the IP address as the destination IP address to connect to the syslog server and initiate the TLS handshake.
3. The SAN field of the incoming certificate from the syslog server is validated using the configured FQDN in the address field. The SAN field contains a list of FQDNs. The following applies:
   - If any FQDN in the SAN list matches the configured FQDN address, the router authenticates the certificate and validates the digital signature.
   - If the SAN list does not contain a match for the FQDN, the router rejects the certificate.

To ensure NIAP compliance, SR OS generates logs for successful and failed SAN verification events.

Example: SAN verification log

```
10 2026/01/09 07:43:03.741 UTC MINOR: TLS #2004 management tls
"TLS session SAN verification status: success, application: syslog, remote addre
ss: www.syslog.sk, TLS Session ID: 3"
```

7.2.8 NETCONF

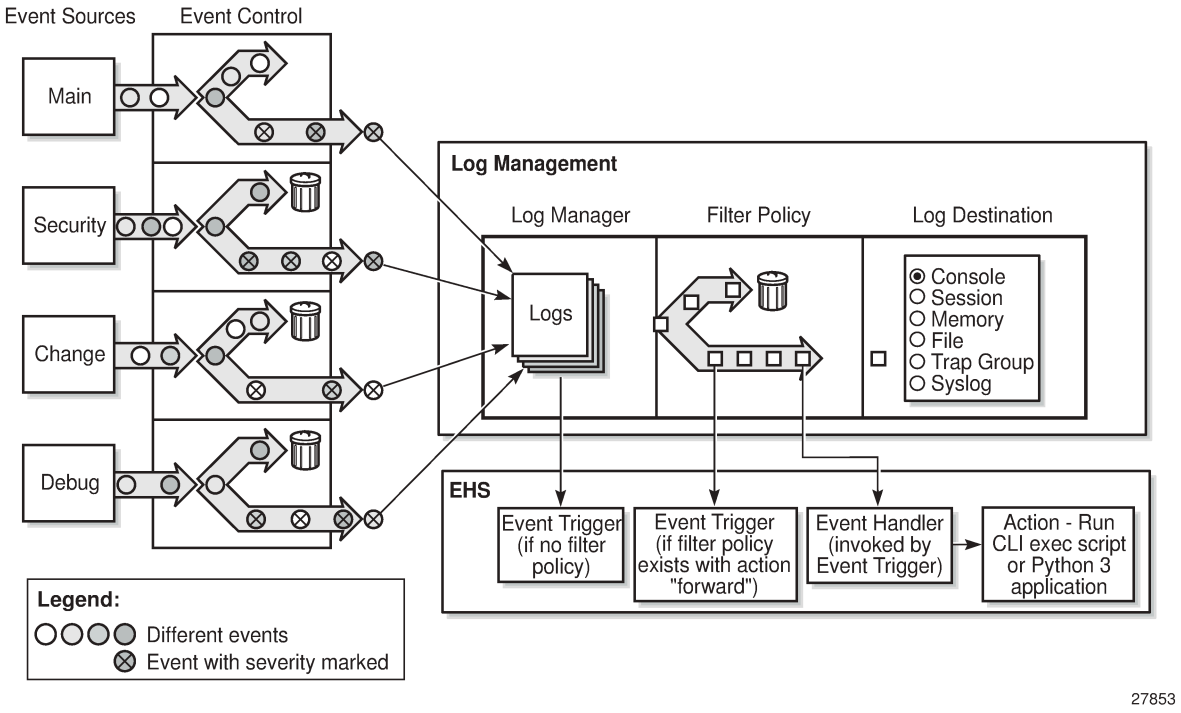
A NETCONF log is a log that outputs log events to a NETCONF session as notifications. A NETCONF client can subscribe to a NETCONF log using the configured **netconf-stream** *stream-name* for the log in a subscription request. See NETCONF notifications for more details.

7.3 Event logs

Event logs are the means of recording system-generated events for later analysis. Events are messages generated by the system by applications or processes within the router.

The following figure shows a function block diagram of event logging.

Figure 21: Event logging block diagram



7.3.1 Event sources

In Figure 21: Event logging block diagram, the event sources are the main categories of events that feed the log manager:

- **security**

  The security event source is all events that affect attempts to breach system security such as failed login attempts, attempts to access MIB tables to which the user is not granted access or attempts to enter a branch of the CLI to which access has not been granted. Security events are generated by the SECURITY application, as well as several other applications (TLS for example).

- **change**

  The change activity event source receives all events that directly affect the configuration or operation of the node. Change events are generated by the USER application. The Change event stream also includes the tmnxConfigModify(#2006), tmnxConfigCreate (#2007), tmnxConfigDelete (#2008), and tmnxStateChange (#2009) change events from the SYSTEM application, as well as the various xxxConfigChange events from the MGMT_CORE application.

- **debug**

  The debug event source is the debugging configuration that has been enabled on the system. Debug events are generated when debug is enabled for various protocols under the **debug** branch of the CLI (for example, **debug system ntp**).

- **li**

  The li event source generates lawful intercept events from the LI application.

- **main**

  The main event source receives events from all other applications within the router.

The event source for a particular log event is displayed in the output of the following command.

```
show log event-control detail
```

The event source can also be found in the **source-stream** element in the **state log log-events** context.

Use the following command to show the list of event log applications.

```
show log applications
```

The following example shows the **show log applications** command output. Examples of event log applications include IP, MPLS, OSPF, CLI, services, and so on.

Output example

```
==================================
Log Event Application Names
==================================
Application Name
----------------------------------
...
BGP
CCAG
CFLOWD
CHASSIS
...
MPLS
MSDP
NTP
...
USER
VRRP
VRTR
==================================
```

7.3.2 Event control

Event control pre-processes the events generated by applications before the event is passed into the main event stream. Event control assigns a severity to application events and can either forward the event to the main event source or suppress the event. Suppressed events are counted in event control, but these events will not generate log entries as they never reach the log manager.

Simple event throttling is another method of event control and is configured similarly to the generation and suppression options. See Simple logger event throttling.

Events are assigned a default severity level in the system, but the application event severities can be changed by the user.

Application events contain an event number and description that describes why the event is generated. The event number is unique within an application, but the number can be duplicated in other applications.

Use the following command to display log event information.

```
show log event-control
```

The following example, generated by querying event control for application generated events, shows a partial list of event numbers and names.

Output example

```
===============================================================================
Log Events
===============================================================================
Application
 ID#    Event Name                          P   g/s     Logged     Dropped
-------------------------------------------------------------------------------

show
BGP:
   2001 bgpEstablished                     MI  gen          1           0
   2002 bgpBackwardTransition              WA  gen          7           0
   2003 tBgpMaxPrefix90                    WA  gen          0           0
...
CCAG:
CFLOWD:
   2001 cflowdCreated                      MI  gen          1           0
   2002 cflowdCreateFailure                MA  gen          0           0
   2003 cflowdDeleted                      MI  gen          0           0
...
CHASSIS:
   2001 cardFailure                        MA  gen          0           0
   2002 cardInserted                       MI  gen          4           0
   2003 cardRemoved                        MI  gen          0           0
...

,,,
DEBUG:
L  2001 traceEvent                         MI  gen          0           0
DOT1X:
FILTER:
   2001 filterPBRPacketsDropped            MI  gen          0           0
IGMP:
   2001 vRtrIgmpIfRxQueryVerMismatch       WA  gen          0           0
   2002 vRtrIgmpIfCModeRxQueryMismatch     WA  gen          0           0
IGMP_SNOOPING:
IP:
L  2001 clearRTMError                      MI  gen          0           0
L  2002 ipEtherBroadcast                   MI  gen          0           0
L  2003 ipDuplicateAddress                 MI  gen          0           0
...
ISIS:
   2001 vRtrIsisDatabaseOverload           WA  gen          0           0
```

7.3.3 Log manager and event logs

Events that are forwarded by event control are sent to the log manager. The log manager manages the event logs in the system and the relationships between the log sources, event logs and log destinations, and log filter policies.

An event log has the following properties:

- A unique log ID that is a short, numeric identifier for the event log. A maximum of ten logs can be configured at one time.

- One or more log source streams that can be sent to specific log destinations. The source must be identified before the destination can be specified. The events can be from the main event stream, in the security event stream, or in the user activity stream.
- A single destination. The destination for the log ID destination can be console, session, syslog, snmp-trap-group, memory, or a file on the local file system.
- An optional event filter policy that defines whether to forward or drop an event or trap based on match criteria.

7.3.4 Event filter policies

The log manager uses event filter policies to allow fine control over which events are forwarded or dropped based on various criteria. Like other policies in the SR OS, filter policies have a default action. The default actions are either:

- Forward
- Drop

Filter policies also include a number of filter policy entries that are identified with an entry ID and define specific match criteria and a forward or drop action for the match criteria.

Each entry contains a combination of matching criteria that define the application, event number, router, severity, and subject conditions. The action for the entry determines how the packets will be treated if they have met the match criteria.

Entries are evaluated in order from the lowest to the highest entry ID. The first matching event is subject to the forward or drop action for that entry.

Valid operators are described in the following table:

Table 38: Valid filter policy operators

Operator	Description
eq	equal to
neq	not equal to
lt	less than
lte	less than or equal to
gt	greater than
gte	greater than or equal to

A match criteria entry can include combinations of:

- Equal to or not equal to a specific system application.
- Equal to or not equal to an event message string or regular expression match.
- Equal to, not equal to, less than, less than or equal to, greater than, or greater than or equal to an event number within the application.

- Equal to, not equal to, less than, less than or equal to, greater than, or greater than or equal to a severity level.
- Equal to or not equal to a router name string or regular expression match.
- Equal to or not equal to an event subject string or regular expression match.

7.3.5 Event log entries

Log entries that are forwarded to a destination are formatted in a way appropriate for the specific destination, whether it is recorded to a file or sent as an SNMP trap, but log event entries have common elements or properties. All application-generated events have the following properties:

- timestamp in UTC or local time
- generating application
- unique event ID within the application
- A router name identifying the router instance that generated the event.
- subject identifying the affected object
- short text description

The general format for an event in an event log with either a memory, console, or file destination is as follows.

```
nnnn <time> TZONE <severity>: <application> #<event-id> <vrtr-name> <subject>
<message>
```

Example: Event log

```
252 2013/05/07 16:21:00.761 UTC WARNING: SNMP #2005 Base my-interface-abc
"Interface my-interface-abc is operational"
```

The specific elements that comprise the general format are described in the following table.

Table 39: Log entry field descriptions

Label	Description
nnnn	The log entry sequence number.
<time>	YYYY/MM/DD HH:MM:SS.SSS
YYYY/MM/DD	The UTC date stamp for the log entry. YYYY — Year MM — Month DD — Date
HH:MM:SS.SSS	The UTC timestamp for the event. HH — Hours (24 hour-format) MM — Minutes

Label	Description
	SS.SSS — Seconds
TZONE	The time zone (for example, UTC, EDT) as configured by the following command. `configure log log-id x time-format`
<severity>	The severity levels of the event: • CRITICAL: a critical severity event • MAJOR: a major severity event • MINOR: a minor severity event • WARNING: a warning severity event • CLEARED: a cleared event • INDETERMINATE: an indeterminate/informational severity event **Note:** The term "INFO" may appear in messages in management interfaces indicating a situation that is less impactful than a "WARNING", or a situation that has an indeterminate impact, but "INFO" is not a log event severity in SR OS.
<application>	The application generating the log message.
<event-id>	The application event ID number for the event.
<vrtr-name>	The router name in a special format used by the logging system (for example, Base or vprn101, where 101 represents the service-id of the VPRN service), representing the router instance that generated the event.
<subject>	The subject/affected object for the event.
message	A text description of the event.

7.3.6 Generating test events

The SR OS can generate test events that have the same message format as real events. The test events consist of fixed and variable parameters, with hyphens indicating the maximum length of the variable parameter, to illustrate the content of an actual log event. This allows customization and testing of the NMS event handling and alarming, ensuring that events are processed correctly. The user can specify an event name or number, or all events, to send as a test event from an application.

Note: Backwards compatibility of the event format with older software releases is not guaranteed, because the contents of an event may be extended with additional information.

Example: Test event for bgpRemoteEndClosedConn (2011)

Use the following command to identify the variable parameters first:

```
show log event-parameters
```

The variable parameter names are indicated in **bold** in the following output. The **subject** parameter in the common event parameters is the only variable parameter. All others are fixed based on the application and system information. The event specific parameters are always variable and are in two formats based on the parameter name.

- Names that are a single word, or words separated with _ (underscore) are a variable length string, for example, **bgp_peer_name**.
- Names that start with a lowercase letter with the word in camelcase are SNMP OIDs, for example, **tBgpASN4Byte**. For more information about these parameters, search for the parameter name in the SNMP MIBs, which can be downloaded from the Nokia Customer Support Portal.

```
[/]
A:admin@node-2# show log event-parameters bgp bgpRemoteEndClosedConn
===============================================================================
Common Event Parameters
        appid
        name
        eventid
        severity
        subject
        gentime
Event Specific Parameters
        bgp_peer_name
        tBgpASN4Byte
===============================================================================
```

The event parameters and message format can also be found in the *7250 IXR, 7450 ESS, 7705 SAR Gen 2, 7750 SR, 7950 XRS, and VSR Log Events Guide*, for example:

```
(ASN $tBgpASN4Byte$) $bgp_peer_name$: remote end closed connection
```

The **subject** and **bgp_peer_name** parameters are strings, and information about **tBgpASN4Byte** can be found in the TIMETRA-BGP-MIB.

```
tBgpASN4Byte                    OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  accessible-for-notify
    STATUS      current
    DESCRIPTION
        "The value of tBgpASN4Byte indicates the autonomous system number for
         the dynamic neighbor."
    ::= { tBgpNotificationObjs 24 }
```

The following command generates the bgpRemoteEndClosedConn test event.

```
tools perform log generate-event bgp bgpRemoteEndClosedConn
```

The following test event is generated by the preceding command, with variable parameters indicated in **bold**.

```
91 2025/05/21 17:39:31.653 EDT WARNING: BGP #2011 Base subject--.---------.---------.—
"(ASN 4254967294) bgp_peer_name------.---------.---------.---------.---------.---------.-------
--.---------.------: remote end closed connection"
```

The corresponding real event where the variable parameters are populated with the relevant information is formatted as follows, with variable parameters indicated in **bold**.

```
92 2025/05/21 17:39:45.244 EDT WARNING: BGP #2011 Base Peer 1: 10.10.1.77
"(ASN 65551) VR 1: Group IBGP Peers: Peer 10.10.1.77: remote end closed connection"
```

7.3.7 Simple logger event throttling

Simple event throttling provides a mechanism to protect event receivers from being overloaded when a scenario causes many events to be generated in a very short period of time. A throttling rate, # events/ # seconds, can be configured. Specific event types can be configured to be throttled. When the throttling event limit is exceeded in a throttling interval, any further events of that type cause the dropped events counter to be incremented.

Use the commands in the following context to display dropped event counts.

```
show log event-control
```

Events are dropped before being sent to one of the logger event collector tasks. There is no record of the details of the dropped events and therefore no way to retrieve event history data lost by this throttling method.

A particular event type can be generated by multiple managed objects within the system. At the point this throttling method is applied the logger application has no information about the managed object that generated the event and cannot distinguish between events generated by object "A" from events generated by object "B". If the events have the same event-id, they are throttled regardless of the managed object that generated them. It also does not know which events may eventually be logged to destination log-id <n> from events that are logged to destination log-id <m>.

Throttle rate applies commonly to all event types. It is not configurable for a specific event-type.

A timer task checks for events dropped by throttling when the throttle interval expires. If any events have been dropped, a TIMETRA-SYSTEM-MIB::tmnxTrapDropped notification is sent.

7.3.8 Default system log

Log 99 is a pre-configured memory-based log which logs events from the main event source (not security, debug, and so on). Log 99 exists by default.

The following example displays the log 99 configuration.

Example: MD-CLI

```
[ex:/configure log]
A:admin@node-2# info
    log-id "99" {
        admin-state enable
        description "Default system log"
        source {
            main true
        }
        destination {
            memory {
                max-entries 500
            }
```

```
        }
    }
    snmp-trap-group "7" {
    }
```

Example: classic CLI

```
A:node-2>config>log# info detail
#------------------------------------------
echo "Log Configuration "
#------------------------------------------
...
        snmp-trap-group 7
        exit
...
        log-id 99
            description "Default system log"
            no filter
            from main
            to memory 500
            no shutdown
        exit
-------------------------------------------
```

7.3.9 Event handling system

The Event Handling System (EHS) is a framework that allows operator-defined behavior to be configured on the router. EHS adds user-controlled programmatic exception handling by allowing the execution of either a CLI script or a Python 3 application when a log event (the "trigger") is detected. Various fields in the log event provide regexp style expression matching, which allows flexibility for the trigger definition.

EHS handler objects are used to tie together the following:

- trigger events (typically log events that match a configurable criteria)
- a set of actions to perform (enabled using CLI scripts and Python applications)

EHS, along with CRON, may execute SR OS CLI scripts or Python 3 applications to perform operator-defined functions as a result of receiving a trigger event. The Python programming language provides an extensive framework for automation activities for triggered or scheduled events, including model-driven transactional configuration and state manipulation. See the Python chapter for more information.

The use of Python applications from EHS is supported only in model-driven configuration mode.

The following figure shows the relationships among the different configurable objects used by EHS (and CRON).

Figure 22: EHS object handling (MD-CLI)

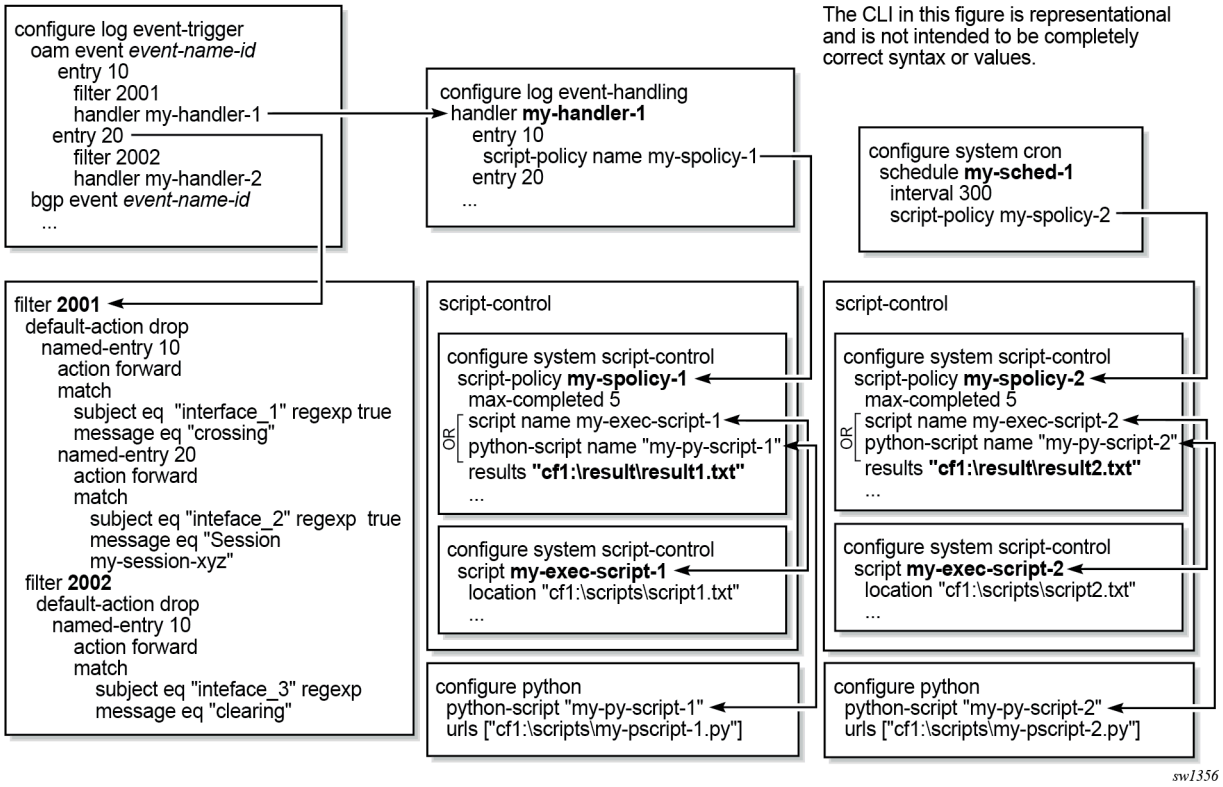
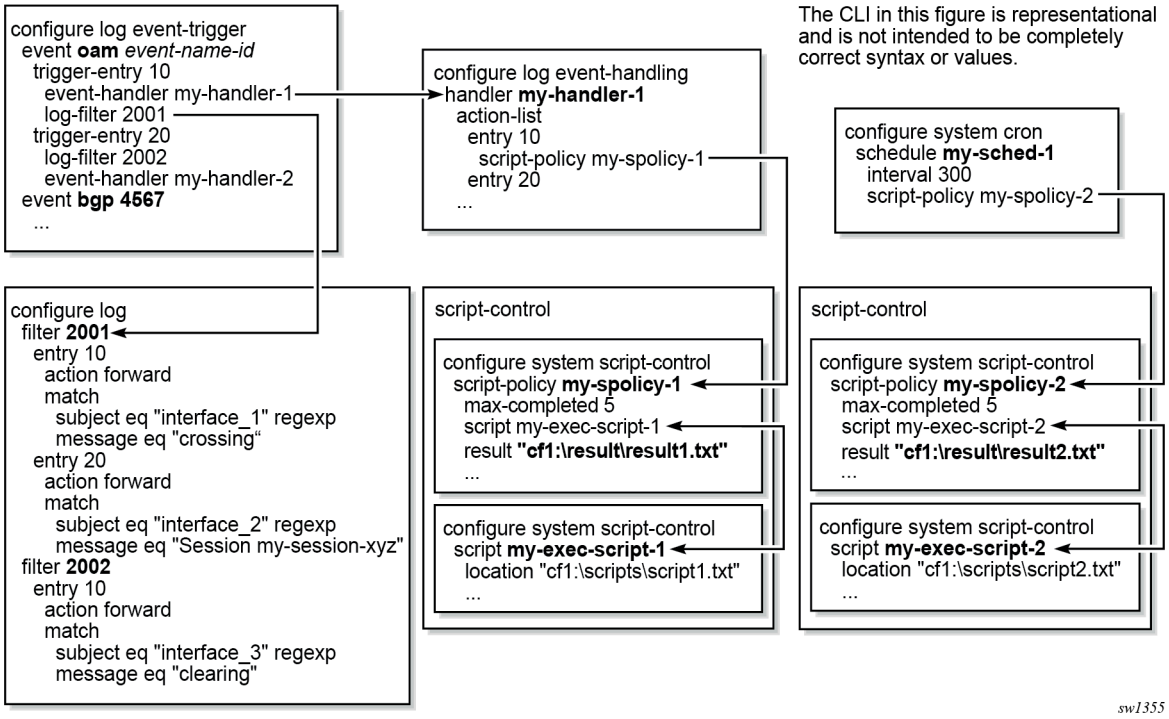


Figure 23: EHS object handling (classic CLI)



7.3.9.1 EHS configuration and variables

You can configure complex rules to match log events as the trigger for EHS. For example, use the commands in the following context to configure discard using suppression and throttling:

- **MD-CLI**

```
configure log log-events
```

- **classic CLI**

```
configure log event-control
```

When a log event is generated in SR OS, it is subject to discard using the configured suppression and throttling before it is evaluated as a trigger for EHS, according to the following:

- EHS does not trigger on log events that are suppressed through the configuration.
- EHS does not trigger on log events that are throttled by the logger.

EHS is triggered on log events that are dropped by user-configured log filters assigned to individual logs.

Use the following command to assign log filters:

```
configure log filter
```

The EHS event trigger logic occurs before the distribution of log event streams into individual logs.

The parameters from the log event are passed into the triggered EHS CLI script or Python application. For CLI scripts, the parameters are passed as individual dynamic variables (for example, $eventid). For Python applications, see the details in the following sections. The parameters are composed of:

- common events
- event specific options

The common event parameters are:

- appid
- eventid
- severity
- gentime (in UTC)
- timestamp (in seconds, available within a Python application only)

The event specific parameters depend on the log event. Use the following command to obtain information for a particular log event.

```
show log event-parameters
```

Alternatively, in the MD-CLI use the following command for information.

```
state log log-events
```

Note: The event sequence number is not passed into the script.

7.3.9.2 Triggering a CLI script from EHS

When using the classic CLI, an EHS script has the ability to define local (static) variables and uses basic .if or .set syntax inside the script. The use of variables with .if or .set commands within an EHS script adds more logic to the EHS scripting and allows the reuse of a single EHS script for more than one trigger or action.

Both passed-in and local variables can be used within an EHS script, either as part of the CLI commands or as part of the .if or .set commands.

The following applies to both CLI commands and .if or .set commands (where *X* represents a variable):

- Using $*X*, without using single or double quotes, replaces the variable *X* with its string or integer value.
- Using "*X*", with double quotes, means the actual string *X*.
- Using "$*X*", with double quotes, replaces the variable *X* with its string or integer value.
- Using '*X*', with single quotes does not replace the variable X with its value but means the actual string $*X*.

The following interpretation of single and double quotes applies:

- All characters within single quotes are interpreted as string characters.
- All characters within double quotes are interpreted as string characters except for $, which replaces the variable with its value (for example, shell expansion inside a string).

7.3.9.2.1 Examples of EHS syntax supported in the classic CLI

This section describes the supported EHS syntax for the classic CLI.

Note: These scenarios use pseudo syntax.

- .if $*string_variable*==*string_value_or_string_variable* {
  CLI_commands_set1
  .} else {
  CLI_commands_set2
  .} endif
- .if ($*string_variable*==*string_value_or_string_variable*) {
  CLI_commands_set1
  .} else {
  CLI_commands_set2
  .} endif
- .if $*integer_variable*==*integer_value_or_integer_variable* {
  CLI_commands_set1
  .} else {
  CLI_commands_set2
  .} endif
- .if ($*integer_variable*==*integer_value_or_integer_variable*) {
  CLI_commands_set1
  .} else {
  CLI_commands_set2
  .} endif
- .if $*string_variable*!=*string_value_or_string_variable* {
  CLI_commands_set1
  .} else {
  CLI_commands_set2
  .} endif
- .if ($*string_variable*!=*string_value_or_string_variable*) {
  CLI_commands_set1
  .} else {
  CLI_commands_set2
  .} endif
- .if $*integer_variable*!=*integer_value_or_integer_variable* {

CLI_commands_set1

.} else {

CLI_commands_set2

.} endif

- .if ($*integer_variable*!=*integer_value_or_integer_variable*) {

  CLI_commands_set1

  .} else {

  CLI_commands_set2

  .} endif
- .set $*string_variable* = *string_value_or_string_variable*
- .set ($*string_variable* = *string_value_or_string_variable*)
- .set $*integer_variable* = *integer_value_or_integer_variable*
- .set ($*integer_variable* = *integer_value_or_integer_variable)*

where:

- *CLI_commands_set1* is a set of one or more CLI commands
- *CLI_commands_set2* is a set of one or more CLI commands
- *string_variable* is a local (static) string variable
- *string_value_or_string_variable* is a string value/variable
- *integer_variable* is a local (static) integer variable
- *integer_value_or_integer_variable* is an integer value/variable

Note:

- A limit of 100 local (static) variables per EHS script is imposed. Exceeding this limit may result in an error and partial execution of the script.
- When a set statement is used to set a string_variable to a string_value, the string_value can be any non-integer value not surrounded by single or double quotes, or it can be surrounded by single or double quotes.
- A "." preceding a directive (for example, if, set...and so on) is always expected to start a new line.
- An end of line is always expected after {.
- A CLI command is always expected to start a new line.
- Passed-in (dynamic) variables are always read-only inside an EHS script and cannot be overwritten using a set statement.
- .if commands support == and != operators only.
- .if and .set commands support the addition, subtraction, multiplication, and division of integers.
- .if and .set commands support the addition of strings, which means concatenation of strings.

7.3.9.2.2 Valid examples for EHS syntax in the classic CLI

This section provides a list of valid examples to trigger log events using EHS syntax in the classic CLI:

- configure service epipe $serviceID

  where *$serviceID* is either a local (static) integer variable or passed-in (dynamic) integer variable
- echo srcAddr is $srcAddr

  where *$srcAddr* is a passed-in (dynamic) string variable
- .set $ipAddr = "10.0.0.1"

  where *$ipAddr* is a local (static) string variable
- .set $ipAddr = $srcAddr

  where *$srcAddr* is a passed-in (dynamic) string variable

  $ipAddr is a local (static) string variable.
- .set ($customerID = 50)

  where *$customerID* is a local (static) integer variable
- .set ($totalPackets = $numIngrPackets + $numEgrPackets)

  where *$totalPackets*, *$numIngrPackets*, *$numEgrPackets* are local (static) integer variables
- .set ($portDescription = $portName + $portLocation)

  where *$portDescription*, *$portName*, *$portLocation* are local (static) string variables
- if ($srcAddr == "CONSOLE") {

  CLI_commands_set1

  .else {

  CLI_commands_set2

  .} endif

  where *$srcAddr* is a passed-in (dynamic) string variable

  CLI_commands_set1 is a set of one or more CLI commands

  CLI_commands_set2 is a set of one or more CLI commands
- .if ($customerId == 10) {

  CLI_commands_set1

  .else {

  CLI_commands_set2

  .} endif

  where *$customerID* is a passed-in (dynamic) integer variable *CLI_commands_set1* is a set of one or more CLI commands

  CLI_commands_set2 is a set of one or more CLI commands
- .if ($numIngrPackets == $numEgrPackets) {

  CLI_commands_set1

  .else {

CLI_commands_set2

.} endif

where *$numIngrPackets* and *$numEgrPackets* are local (static) integer variables

CLI_commands_set1 is a set of one or more CLI commands

CLI_commands_set2 is a set of one or more CLI commands

7.3.9.2.3 Invalid examples for EHS syntax in the classic CLI

This section provides a list of invalid variable use in EHS syntax in the classic CLI:

- .set $srcAddr = "10.0.0.1"

  where *$srcAddr* is a passed-in (dynamic) string variable

  Reason: passed-in variables are read only inside an EHS script.
- .set ($ipAddr = $numIngrPackets + $numEgrPackets)

  where *$ipAddr* is a local (static) string variable

  $numIngrPackets and *$numEgrPackets* are local (static) integer variables

  Reason: variable types do not match, cannot assign a string to an integer.
- .set ($numIngrPackets = $ipAddr + $numEgrPackets)

  where *$ipAddr* is a local (static) string variable

  $numIngrPackets and *$numEgrPackets* are local (static) integer variables

  Reason: variable types do not match, cannot concatenate a string to an integer.
- .set $ipAddr = "10.0.0.1"100

  where *$ipAddr* is a local (static) string variable

  Reason: when double quotes are used, they have to surround the entire string.
- .if ($totalPackets == "10.1.1.1") {

  .} endif

  where *$totalPackets* is a local (static) integer variables

  Reason: cannot compare an integer variable to a string value.
- .if ($ipAddr == 10) {

  .} endif

  where *$ipAddr* is a local (static) string variable

  Reason: cannot compare a string variable to an integer value.
- .if ($totalPackets == $ipAddr) {

  where *$totalPackets* is a local (static) integer variables

  $ipAddr is a local (static) string variable

  Reason: cannot compare an integer variable to a string variable.

7.3.9.3 Triggering a Python application from EHS

When using model-driven configuration mode and the MD-CLI, EHS can trigger a Python application that is executed inside a Python interpreter running on SR OS. See the Python chapter for more information.

Python applications are not supported in classic configuration mode or mixed configuration mode.

When developing an EHS Python application, the event attributes are passed to the application using the **get_event** function in the **pysros.ehs** module.

To import this module, the Python application developer must add the following statement to the application.

```
from pysros.ehs import get_event
```

Use the **get_event** function call to obtain the event triggered the Python application to run. The following example catches the event and returns a Python object into the event variable.

```
event = get_event()
```

When using an EHS Python application, the operator can use the Python programming language to create applications, as required. See the Python chapter for information about displaying model-driven state or configuration information, performing transactional configuration of SR OS, or executing CLI commands in Python.

Common event parameters (group one) are available in Python from the object created using the **get_event** function, as shown in the following table (the functions assume that the EHS event object is called **event**).

Table 40: Python get_event common parameters

Function call	Description	Example output	Python return type
event.appid	The name of the application that generated the event	SYSTEM	String
event.eventid	The event ID number of the application	2068	Integer
event.severity	The severity level of the event	minor	String
event.subject	The subject or affected object of the event	EHS script	String
event.gentime	The formatted time the event was generated in UTC	The timestamp in ISO 8601 format (consistent with state date/time leaves) that the event was generated. For example, 2021-03-08T11:52:06.0-05:00	String

Function call	Description	Example output	Python return type
event.timestamp	The timestamp that the event was generated (in seconds)	1632165026.921208	Float

The variable parameters (group two) are available in Python in the eventparameters attribute of the event object, as shown in the following table. They are presented as a Python dictionary (unordered).

Table 41: Variable parameters available in Python

Function call	Description	Example output	Python return type
event.eventparameters	The event specific variable parameters	<EventParams> When calling keys() on this object the example output is: ('tmnxEhsHandlerName', 'tmnxEhsHEntryId', 'tmnx EhsHEntryScriptPlcyOwner', 'tmnxEhsHEntryScript PlcyName', 'smLaunch Owner', 'smLaunchName', 'smLaunchScriptOwner', 'smLaunchScriptName', 'smLaunchError', 'tmnx SmLaunchExtAuthType', 'sm RunIndex', 'tmnxSmRunExt AuthType', 'tmnxSmRunExt UserName')	Dict

In addition to the variables, the **format_msg()** function is provided to output the formatted log string from the event as it would appear in the output of the **show log** command.

Example: format_msg() usage

```
print(event.format_msg())
```

Example: Output of the format_msg() function

```
Launch of none operation failed with a error: Python script's operational status is not
 'inService'. The script policy "test_ehs" created by the owner "TiMOS CLI" was executed
 with cli-user account "not-specified"
```

7.3.9.4 EHS debounce

EHS debounce (also called dampening) is the ability to trigger an action (for example an EHS script), if an event happens (N) times within a specific time window (S).

N = [2..15]

S = [1..604800]

Note:

- Triggering occurs with the Nth event, not at the end of S.
- There is no sliding window (for example a trigger at Nth event, N+1 event, and so on), because N is reset after a trigger and the count is restarted.
- When EHS debouncing or dampening is used, the varbinds passed in to an EHS script at script triggering time are from the Nth event occurrence (the Nth triggering event).
- If S is not specified, the SR OS continues to trigger every Nth event.

For example, when linkDown occurs N times in S seconds, an EHS script is triggered to shut down the port.

7.3.9.5 Executing EHS or CRON CLI scripts or Python applications

The execution of EHS or CRON scripts depends on the CLI engine associated with the configuration mode. The EHS or CRON script execution engine is based on the configured primary CLI engine. Use the following command to configure the primary CLI engine.

```
configure system management-interface cli cli-engine
```

For example, if **cli-engine** is configured to **classic-cli**, the script executes in the classic CLI infrastructure and disregards the configuration mode, even if it is model-driven.

Note: Configuration changes made with CLI scripts must be saved with the **admin save** command, regardless of the **auto-config-save** settings. Configuration changes made with Python applications are saved according to the NETCONF **auto-config-save** setting except when executed by CRON or EHS when they are not automatically saved.

The following is the default behavior of the EHS or CRON scripts, depending on the configuration mode:

- **model-driven configuration mode**

  EHS or CRON scripts execute in the MD-CLI environment and an error occurs if any classic CLI commands exist. Python applications are fully supported and use the SR OS model-driven interfaces and the pySROS libraries to obtain and manipulate state and configuration data, as well as pySROS API calls to execute MD-CLI commands.

- **classic configuration mode**

  EHS or CRON scripts execute in the classic CLI environment and an error occurs if any MD-CLI commands exist. Python applications are not supported and the system returns an error.

- **mixed configuration mode**

  EHS or CRON scripts execute in the classic CLI environment and an error occurs if any MD-CLI commands exist. Python applications are not supported and the system returns an error.

EHS or CRON scripts that contain MD-CLI commands can be used in the MD-CLI as follows:

- scripts can be configured
- scripts can be created, edited, and results read through FTP
- scripts can be triggered and executed
- scripts generate an error if there are any non MD-CLI commands or .if or .set syntax in the script

Use the following commands to configure user authorization for EHS or CRON scripts and Python applications:

- **MD-CLI and classic CLI**

```
configure system security cli-script authorization event-handler cli-user
configure system security cli-script authorization cron cli-user
```

- **MD-CLI only**

```
configure system security python-script authorization event-handler cli-user
configure system security python-script authorization cron cli-user
```

When a user is not specified, an EHS or CRON script and an EHS or CRON Python application bypasses authorization and can execute all commands.

In all configuration modes, a script policy can be disabled using the following command even if history exists:

- **MD-CLI**

```
configure system script-control script-policy admin-state disable
```

- **classic CLI**

```
configure system script-control script-policy shutdown
```

When the script policy is disabled, the following applies:

- Newly triggered EHS or CRON scripts or Python applications are not allowed to execute or queue.
- In-progress EHS or CRON scripts or Python applications are allowed to continue.
- Already queued EHS or CRON scripts or Python applications are allowed to execute.

By default, a script policy is configured to allow an EHS or CRON script to override datastore locks from any model-driven interface (MD-CLI, NETCONF, and so on) in mixed and model-driven modes. Use the following command to configure a script policy to prevent EHS or CRON scripts from overriding datastore locks:

- **MD-CLI**

```
configure system script-control script-policy lock-override false
```

- **classic CLI**

```
configure system script-control script-policy no lock-override
```

7.3.10 Managing logging in VPRNs

Log events can be sent from within a VPRN instead of from the base router instance or the CPM management router instance. For example, a syslog collector may be reachable through a VPRN interface.

To deploy VPRN logs, the user must configure an event log inside the following context.

```
configure service vprn log
```

By default, the event source streams for VPRN event logs contain only events that are associated with the specific VPRN. To send a VPRN event log for the entire system-wide set of log events (VPRN and non-VPRN), use the following command. This can be useful, for example, when a VPRN is being used as a management VPRN.

```
configure log services-all-events
```

7.4 Custom log events

The SR OS supports six custom log events, each with a different default severity that is modifiable like any other log event. The event names and associated default severity of the events are described in the following table.

Table 42: Custom log events and severities

Event name	Default severity
tmnxCustomEvent1	critical
tmnxCustomEvent2	major
tmnxCustomEvent3	minor
tmnxCustomEvent4	warning
tmnxCustomEvent5	cleared
tmnxCustomEvent6	indeterminate

A custom event can be raised by a user or client with the **perform log custom-event** command in MD-CLI, a YANG modeled operation NETCONF, or a pySROS script. The subject, message text, and multiple output parameters of the log event can be populated with custom strings.

The custom log events can be used as triggers for Event Handling System (EHS) handlers, with all parameters passed into the associated EHS scripts. The events can also be sent to any standard log destination type, for example syslog or SNMP notifications.

Example: Custom generic log event raised in MD-CLI

The following is an example of a custom generic log event raised in MD-CLI.

```
[/]
A:admin@node-2# perform log custom-event 4 subject "test" message-string "Port 1/1/1 is in
 the Down state" parameter1 "1/1/1" parameter2 "Down" parameter3 "1977-05-04"
```

The resulting log event is shown in the **show log log-id 99** command.

```
[/]
A:admin@node-2# show log log-id 99
===============================================================================
Event Log 99 log-name 99
===============================================================================
Description : Default System Log
Memory Log contents  [size=500   next event=45  (not wrapped)]
```

```
44 2024/05/02 12:21:12.423 UTC WARNING: LOGGER #2023 Base test "Port 1/1/1 is in the Down
 state"
```

If the log event in the preceding example is sent as an SNMP notification, or if the log event is used as a trigger for the EHS, the following event-specific parameters are passed:

- logCustomEventSubject = “test”
- logCustomEventMessageString = “Port 1/1/1 is in the Down state”
- logCustomEventParameter1 = “1/1/1”
- logCustomEventParameter2 = “Down”
- logCustomEventParameter3 = “1977-05-04”

The total length of the **message-string** plus all parameters (**parameter1** to **parameter8**) strings must be equal to or less than 2400 characters.

Embedded double quotes are supported in the **message-string** and parameter inputs by using the backslash character (\) followed immediately by the double quote character (").

Example: Custom log event with double quotes

The following is an example configuration of a custom log event.

```
[/]
A:admin@node-2# perform log custom-event 1 subject "test" message-string "{
\"nokia-conf:connect-retry\": 90,  \"nokia-conf:local-preference\": 250,
\"nokia-conf:add-paths\": { \"ipv4\": {  \"receive\": true } }}"
```

The following string is the output of the message field of the resulting log event.

```
"{ "nokia-conf:connect-retry": 90, "nokia-conf:local-preference": 250,
"nokia-conf:add-paths": { "ipv4": { "receive": true } }}"
```

7.5 Test log event

The SR OS provides the tmnxTestEvent test event with optional custom text. The test event can be generated with the **perform log test-event** command. The **custom-text** command in this context replaces the default message of the event.

The total length of the **custom-text** must be equal to or less than 800 characters. Embedded double quotes are not supported in the **custom-text** string. There is no special treatment for \n or \r sequences. For example, \n in the **custom-text** string is output as the backslash character (\) and “n” (the equivalent of ASCII 0x5C and 0x6e).

Example: Test event with the default message

```
[/]
A:admin@node-2# perform log test-event
```

The following test event is generated by the preceding command.

```
2728 2025/05/20 16:29:40.636 EDT INDETERMINATE: LOGGER #2011 Base Event Test
"Test event has been generated with system object identifier tmnxModelSR1v2Reg
System description: TiMOS-B-25.3.R2 both/x86_64 Nokia 7750 SR Copyright (c) 2000-2025
```

```
 Nokia.
All rights reserved. All use subject to applicable license agreements.
Built on Wed Apr 23 20:04:18 UTC 2025 by builder in /builds/253B/R2/panos/main/sros
"
```

Example: Test event with custom text

```
[/]
A:admin@node-2# perform log test-event custom-text "Starting maintenance window 7728\n\r
 Now"
```

The following test event is generated by the preceding command.

```
2731 2025/05/20 16:30:46.950 EDT INDETERMINATE: LOGGER #2011 Base Event Test
"Starting maintenance window 7728\n\r Now"
```

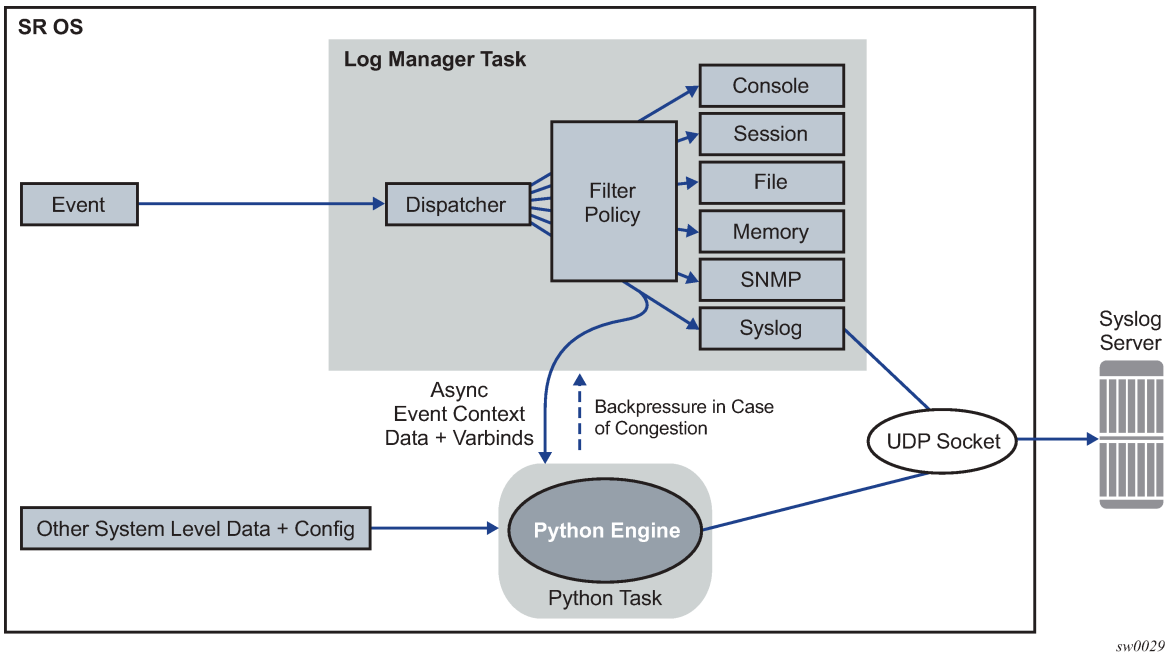
7.6 Customizing Syslog messages using Python

Note: The Python 3 pySROS modules (except pysros.syslog) are not available for use with Syslog message customization.

Note: The Python syslog customization feature does not support SR OS filesystem access from Python.

Any log events in SR OS can be customized using a Python script before they are sent to a syslog server. If the result of a log filter is to drop the event, no further processing occurs and the message is not sent. The following figure shows the interaction between the logger and the Python engine.

Figure 24: Interaction between the logger and the Python engine



7.6.1 Python engine for syslog

This section describes the syslog-specific aspects of Python processing.

When an event is dispatched to the log manager in SR OS, the log manager asynchronously passes the event context data and variables (varbinds in Python 2 and event parameters in Python 3) to the Python engine; that is, the logger task is not waiting for feedback from Python.

Varbinds or event parameters are variable bindings that represent the variable number of values that are included in the event. Each varbind in Python 2 consists of a triplet (OID, type, value).

Along with other system-level variables, the Python engine constructs a Syslog message and sends it to the syslog destination when the Python engine successfully concludes. During this process, the operator can modify the format of the Syslog message or leave it intact, as if it was generated by the syslog process within the log manager.

The tasks of the Python engine in a syslog context are as follows:

- assemble custom Syslog messages (including PRI, HEADER and MSG fields) based on the received event context data, varbinds and event parameters specific to the event, system-level data, and the configuration parameters (syslog server IP address, syslog facility, log-prefix, and the destination UDP port)
- reformat timestamps in a Syslog message
- modify attributes in the message and reformats the message
- send the original or modified message to the syslog server
- drop the message

7.6.1.1 Python 2 syslog APIs

Python APIs are used to assemble a Syslog message which, in SR OS, has the format described in section Syslog.

The following table describes Python information that can be used to manipulate Syslog messages.

Table 43: Manipulating Python Syslog messages

Imported Nokia (ALC) modules	Access rights	Comments
event (from alc import event)	—	Method used to retrieve generic event information
syslog (from alc import syslog)	—	Method used to retrieve syslog-specific parameters
system (from alc import system)	—	Method used to retrieve system-specific information. Currently, the only parameter retrieved is the system name.
Events use the following format as they are written into memory, file, console, and system: nnnn <time> <severity>:<application> # <event_id> <router-name> <subject> <message>		

Imported Nokia (ALC) modules	Access rights	Comments
The event-related information received in the context data from the log manager is retrieved via the following Python methods:		
event.sequence	RO	Sequence number of the event (nnnn)
event.timestamp	RO	Event timestamp in the format: (YYYY/MM/DD HH:MM:SS.SS)
event.routerName	RO	Router name, for example, BASE, VPRN1, and so on
event.application	RO	Application generating the event, for example, NA
event.severity	RO	Event severity configurable in SR OS (CLEARED [1], INFO [2], CRITICAL [3], MAJOR [4], MINOR [5], WARNING [6]).
event.eventId	RO	Event ID; for example, 2012
event.eventName	RO	Event Name; for example, tmnxNatPlBlocl AllocationLsn
event.subject	RO	Optional field; for example, [NAT]
event.message	RO	Event-specific message; for example, "{2} Map 192.168.20.29 [2001-2005] MDA 1/ 2 -- 276824064 classic-lsn-sub %3 vprn1 10.10.10.101 at 2015/08/31 09:20:15"
Syslog methods		
syslog.hostName	RO	IP address of the SR OS node sending the Syslog message. This is used in the Syslog HEADER.
syslog.logPrefix	RO	Log prefix which is configurable and optional; for example, TMNX:
syslog.severityToPRI(event.severity)	—	Python method used to derive the PRI field in syslog header based on event severity and a configurable syslog facility
syslog.severityToName(event.severity)	—	SR OS event severity to syslog severity name. For more information, see the Syslog section.
syslog.timestampToUnix(timestamp)	—	Python method that takes a timestamp in the YYYY/MM/DD HH:MM:SS format and converts it into a UNIX-based format (seconds from Jan 01 1970 – UTC)
syslog.set(newSyslogPdu)	—	Python method used to send the Syslog message in the newSyslogPdu. This variable

Imported Nokia (ALC) modules	Access rights	Comments
		must be constructed manually via string manipulation. In the absence of the command, the SR OS assembles the default Syslog message (as if Python was not configured) and sends it to the syslog server, assuming that the message is not explicitly dropped.
syslog.drop()	—	Python method used to drop a Syslog message. This method must be called before the syslog.set<newSyslogPdu method.
System methods		
system.name	RO	Python method used to retrieve the system name

For example, assume that the syslog format is:

```
<PRI><timestamp> <hostname> <log-prefix>: <sequence> <router-name>  <appid>-
<severity>-<name>-<eventid> [<subject>]: <text>
```

Then the syslogPdu is constructed via Python as shown in the following example:

```
syslogPdu = "<" + syslog.severityToPRI(event.severity) + ">" \ + event.timestamp + "
 " \ + syslog.hostname + " " + syslog.logPrefix + ": " + \ event.sequence + " " + ev
ent.routerName + " " + \  event.application + "-
" + \ syslog.severityToName(event.severity) + "-" + \
               event.eventName + "-" + event.eventId + " [" + \
               event.subject + "]: " + event.message
```

7.6.1.2 Python 3 syslog APIs

Python APIs are used to modify and assemble a Syslog message which, in SR OS, has the format described in section Syslog.

The syslog module for Python 3 is included in the pySROS libraries pre-installed on the SR OS device. The **get_event** function must be imported from the **pysros.syslog** module at the beginning of each Python 3 application by including the following:

```
from pysros.sylog import get_event
```

The specific event that the syslog handler is processing can be returned in a variable using the following example Python 3 code:

```
my_event = get_event()
```

In the preceding example, my_event is an object of type Event. The Event class provides a number of parameters and functions as described in the following table:

Table 44: Parameters and functions for the Event class

Key name	Python type	Read-only	Description
name	String	N	Event name
appid	String	N	Name of application that generated the log message
eventid	Integer	N	Event ID number of the application
severity	String	N	Severity level of the event (lowercase). The accepted values in SR OS are: • none • cleared • indeterminate • critical • major • minor • warning
sequence	Integer	N	Sequence number of the event in the **syslog** collector
subject	String	N	Subject or affected object for the event
router_name	String	N	Name of the SR OS **router-instance** (for example, Base) in which the event is triggered
gentime	String	Y	Timestamp in ISO 8601 format for the generated event. Example: 2021-03-08T11:52:06.0-0500. Changes to the timestamp field are reflected in this field
timestamp	Float	N	Timestamp, in seconds
hostname	String	N	Hostname field of the Syslog message. This can be an IP address, a fully-qualified domain name, or a hostname.
log_prefix	String	N	Optional log prefix, for example, TMNX
facility	Integer	N	Syslog facility [0-31]
text	String	N	String representation of the text portion of the message only. By default, this is generated from the **eventparameters** attribute.
eventparameters	Dict	Y	Python class that behaves similarly to a Python dictionary of all key, value pairs for all log event specific information that does not fall into the standard fields.

Key name	Python type	Read-only	Description
format_msg()	String	n/a	Formatted version of the full log message as it appears in **show log** **Note: format_msg()** is a function itself and must be called to generate the formatted message.
format_syslog_msg()	String	n/a	Formatted version of the Syslog message as it would be sent to the syslog server. **Note: format_syslog_msg()** is a function itself and must be called to generate the formatted message.
override_payload(*payload***)**		n/a	Provide a custom **syslog** message as it would appear in the packet, including the header information (facility, timestamp, and so on) and body data (the actual message). Attributes from this Event are used to construct a completely new message format. Any prior changes to the values of these attributes are used.
drop()		n/a	Drop the message from the pipeline. The Syslog message is not sent out (regardless of any subsequent changes in the Python script). The script continues normally.

The parameter values for the specific event are provided in the Event class. At the end of the Python application execution, the resultant values are returned to the syslog system to transmit the Syslog message. Any changes made to the read-write parameters are used in the Syslog message unless the **drop()** method is called.

More information about the pysros.syslog module can be found in the API documentation for pySROS delivered with the pySROS libraries.

7.6.1.3 Timestamp format manipulation in Python 2

Certain logging environments require customized formatting of the timestamp. Nokia provides a timestamp conversion method in the alu.syslog Python module to convert a timestamp from the format YYYY/MM/DD hh:mm:ss into a UNIX-based timestamp format (seconds from Jan 01 1970 – UTC).

For example, an operator can use the following Python method to convert a timestamp from the YYYY/MM/DD hh:mm:ss.ss or YYYY/MM/DD hh:mm:ss (no centiseconds) format into either the UNIX timestamp format or the MMM DD hh:mm:ss format.

```
from alc import event
from alc import syslog
from alc import system
#input format: YYYY/MM/DD hh:mm:ss.ss  or YYYY/MM/DD hh:mm:ss
```

```
#output format 1: MMM DD hh:mm:ss
#output format 2: unixTimestamp (TBD)
def timeFormatConversion(timestamp,format):
    if format not in range(1,2):
        raise NameError('Unexpected format, expected:' \
                        '0<format<3 got: '+str(format))
    try:
        dat,tim=timestamp.split(' ')
    except:
        raise NameError('Unexpected timestamp format, expected:' \
                        'YYYY/MM/DD hh:mm:ss got: '+timestamp)
    try:
        YYYY,MM,DD=dat.split('/')
    except:
        raise NameError('Unexpected timestamp format, expected:' \
                        'YYYY/MM/DD hh:mm:ss got: '+timestamp)
    try:
        hh,mm,ss=tim.split(':')
        ss=ss.split('.')[0]   #just in case that the time format is hh:mm:ss.ss
    except:
        raise NameError('Unexpected timestamp format, expected:' \
                        'YYYY/MM/DD hh:mm:ss got: '+timestamp)
    if not (1970<=int(YYYY)<2100 and
            1<=int(MM)<=12 and
            1<=int(DD)<=31 and
            0<=int(hh)<=24 and
            0<=int(mm)<=60 and
            0<=int(ss)<=60):
        raise NameError('Unexpected timestamp format, or values out of the range' \
                        'Expected: YYYY/MM/DD hh:mm:ss got: '+timestamp)
    if format == 1:
        MMM={1:'Jan',
             2:'Feb',
             3:'Mar',
             4:'Apr',
             5:'May',
             6:'Jun',
             7:'Jul',
             8:'Aug',
             9:'Sep',
             10:'Oct',
             11:'Nov',
             12:'Dec'}[int(MM)]
        timestamp=MMM+' '+DD+' '+hh+':'+mm+':'+ss
    if format == 2:
        timestamp=syslog.timestampToUnix(timestamp)
    return timestamp
```

The timeFormatConversion method can accept the event.timestamp value in the format:

```
YYYY/MM/DD HH:MM:SS.SS
```

and return a new timestamp in the format determined by the format parameter:

```
1 # MMM DD HH:MM:SS
2 # Unix based time format
```

This method accepts the input format in either of the two forms, YYYY/MM/DD HH:MM:SS.SS or YYYY/MM/DD HH:MM:SS, and ignores the centisecond part in the former form.

7.6.1.4 Timestamp format manipulation in Python 3

Certain logging environments require customized formatting of the timestamp. The Python 3 interpreter provided with SR OS also provides the **utime** and **datetime** modules for format manipulation.

7.6.2 Python processing efficiency

Python retrieves event-related variables from the log manager, as opposed to retrieving pre-assembled Syslog messages. This eliminates the need for string parsing of the Syslog message to manipulate it constituent parts increasing the speed of Python processing.

To further improve processing performance, Nokia recommends performing string manipulation via the Python native string method, when possible.

7.6.3 Python backpressure

A Python task assembles Syslog messages based on the context information received from the logger and sends them to the syslog server independent of the logger. If the Python task is congested because of a high volume of received data, the backpressure should be sent to the ISA so that the ISA stops allocating NAT resources. This behavior matches the current behavior in which NAT resources allocation is blocked if that logger is congested.

7.6.4 Selecting events for Python processing

About this task

Events destined for Python processing are configured through a log ID that references a Python policy. Event selection is performed using a filter associated with the log ID. The remainder of the events destined for the same syslog server can bypass Python processing by redirecting them to a different log ID.

Procedure

Step 1. Use the commands in the following contexts to create the Python policy and log ID:

- **MD-CLI**

```
configure python python-policy PyForLogEvents
configure python python-policy syslog
```

- **classic CLI**

```
configure python python-policy PyForLogEvents create
configure python python-policy syslog
```

Step 2. Use log filters to identify the events that are subject to Python processing.

Example

MD-CLI

```
[ex:/configure log]
A:admin@node-2# info
```

```
    filter "6" {
        default-action drop
        named-entry "1" {
            action forward
            match {
                application {
                    eq nat
                }
                event {
                    eq 2012
                }
            }
        }
    }
    filter "7" {
        default-action forward
        named-entry "1" {
            action drop
            match {
                application {
                    eq nat
                }
                event {
                    eq 2012
                }
            }
        }
    }
```

Example

classic CLI

```
A:node-2>config>log# info
----------------------------------------------
        filter 6
            default-action drop
            entry 1
                action forward
                match
                    application eq "nat"
                    number eq 2012
                exit
            exit
        exit
        filter 7
            default-action forward
            entry 1
                action drop
                match
                    application eq "nat"
                    number eq 2012
                exit
            exit
        exit
```

Step 3. Specify the syslog destination.

Example

MD-CLI

```
[ex:/configure log]
A:admin@node-2# info
```

```
    syslog "1" {
        address 192.168.1.1
    }
```

Example

classic CLI

```
A:node-2>config>log># info
----------------------------------------------
        syslog 1
            address 192.168.1.1
        exit
```

Step 4. Apply the Python syslog policy to selected events using the specified filters.

In the following example, the configuration-only event 2012 from application "nat" is sent to log-id 33. All other events are forwarded to the same syslog destination using log-id 34, without any modification. As a result, all events (modified using log-id 33 and unmodified using log-id 34) are sent to the syslog 1 destination.

This configuration may cause reordering of Syslog messages at the syslog 1 destination because of slight delay of messages processed by Python.

Example

MD-CLI

```
[ex:/configure log]
A:admin@node-2# info
    log-id "33" {
        admin-state enable
        python-policy "PyForLogEvents"
        filter "6"
        source {
            main true
        }
        destination {
            syslog "1"
        }
    }
    log-id "34" {
        admin-state enable
        filter "7"
        source {
            main true
        }
        destination {
            syslog "1"
        }
    }
```

Example

classic CLI

```
A:node-2>config>log># info
----------------------------------------------
    log-id 33
            filter 6
            from main
            to syslog 1
            python-policy "PyForLogEvents"
```

```
            no shutdown
        exit
        log-id 34
            filter 7
            from main
            to syslog 1
            no shutdown
        exit
```

7.7 Accounting logs

Before an accounting policy can be created, a target log file policy must be created to collect the accounting records. The files are stored in system memory on compact flash (`cf1:` or `cf2:`) in a compressed (tar) XML format and can be retrieved using FTP or SCP.

A file policy can only be assigned to either one event log or one accounting log.

7.7.1 Accounting records

An accounting policy must define a record name and collection interval. Only one record name can be configured per accounting policy. Also, a record name can only be used in one accounting policy.

The record name, sub-record types, and default collection period for some service and network accounting policies are shown in Table 45: Accounting record name and collection periods, Table 47: Policer stats field descriptions, Table 48: Queue group record types, and Table 49: Queue group record type fields provide field descriptions.

Table 45: Accounting record name and collection periods

Record name	Sub-record types	Accounting object	Default collection period (minutes)
service-ingress-octets	sio	SAP	5
service-egress-octets	seo	SAP	5
service-ingress-packets	sip	SAP	5
service-egress-packets	sep	SAP	5
network-ingress-octets	nio	Network port	15
network-egress-octets	neo	Network port	15
network-egress-packets	nep	Network port	15
network-ingress-packets	nio	Network port	15
compact-service-ingress-octets	ctSio	SAP	5

Record name	Sub-record types	Accounting object	Default collection period (minutes)
combined-service-ingress	cmSipo	SAP	5
combined-network-ing-egr-octets	cmNio & cmNeo	Network port	15
combined-service-ing-egr-octets	cmSio & cmSeo	SAP	5
complete-network-ing-egr	cpNipo & cpNepo	Network port	15
complete-service-ingress-egress	cpSipo & cpSepo	SAP	5
combined-sdp-ingress-egress	cmSdpipo and cm Sdpepo	SDP and SDP binding	5
complete-sdp-ingress-egress	cmSdpipo, cm Sdpepo, cpSdpipo and cpSdpepo	SDP and SDP binding	5
complete-subscriber-ingress-egress	cpSBipo & cpSBepo	Subscriber profile	5
aa-protocol	aaProt	AA ISA Group	15
aa-application	aaApp	AA ISA Group	15
aa-app-group	aaAppGrp	AA ISA Group	15
aa-subscriber-protocol	aaSubProt	Special study AA subscriber	15
aa-subscriber-application	aaSubApp	Special study AA subscriber	15
custom-record-aa-sub	aaSubCustom	AA subscriber	15
combined-mpls-lsp-egress	mplsLspEgr	LSP	5
combined-mpls-lsp-ingress	mplsLspIn	LSP	5
saa	saa png trc hop	SAA or SAA test	5
complete-ethernet-port	enet	Ethernet port	15
combined-mpls-srte-egress	mplsSrteEgr	LSP	5
combined-sr-policy-egress	srPolEgr	LSP	5

When creating accounting policies, one service accounting policy and one network accounting policy can be defined as default. If statistics collection is enabled on a SAP or network port and no accounting policy is applied, the respective default policy is used. If no default policy is defined, no statistics are collected unless a specifically defined accounting policy is applied.

Each accounting record name is composed of one or more sub-records, which is, in turn, composed of multiple fields.

See the AA statistics fields generated per record table in the *7705 SAR Gen 2 Multiservice ISA and ESA Guide* for field names for AA records.

See the OAM-PM XML keywords and MIB reference table in the *7705 SAR Gen 2 OAM and Diagnostics Guide* for field names for OAM records.

The following table lists the accounting record name details. The availability of the records listed in the table depends on the specific platform functionality and user configuration.

Table 46: Accounting record name details

Record name	Sub-record	Field	Field description
Service-ingress-octets (sio)	sio	svc	SvcId
		sap	SapId
		host-port	Associated satellite host port ID (optional)[16]
		qid	QueueId
		hoo	OfferedHiPrioOctets
		hod	DroppedHiPrioOctets
		loo	LowOctetsOffered
		lod	LowOctetsDropped
		uco	UncoloredOctetsOffered
		iof	InProfileOctetsForwarded
		oof	OutOfProfileOctetsForwarded
Service-egress-octets (seo)	seo	svc	SvcId
		sap	SapId
		host-port	Associated satellite host port ID (optional)[16]
		qid	QueueId
		iof	InProfileOctetsForwarded
		iod	InProfileOctetsDropped
		oof	OutOfProfileOctetsForwarded

16 The host-port field is only included if the SAP is bound to an Ethernet satellite client port or a LAG with satellite client ports.

Record name	Sub-record	Field	Field description
		ood	OutOfProfileOctetsDropped
Service-ingress-packets (sip) [17]	sip	svc	SvcId
		sap	SapId
		host-port	Associated satellite host port ID (optional)[16]
		qid	QueueId
		hpo	HighPktsOffered
		hpd	HighPktsDropped
		lpo	LowPktsOffered
		lpd	LowPktsDropped
		ucp	UncoloredPacketsOffered
		ipf	InProfilePktsForwarded
		opf	OutOfProfilePktsForwarded
Service-egress-packets (sep) [17]	sep	svc	SvcId
		sap	SapId
		host-port	Associated satellite host port ID (optional)[16]
		qid	QueueId
		ipf	InProfilePktsForwarded
		ipd	InProfilePktsDropped
		opf	OutOfProfilePktsForwarded
		opd	OutOfProfilePktsDropped
Network-ingress-octets (nio)	nio	port	PortId
		qid	QueueId
		iof	InProfileOctetsForwarded
		iod	InProfileOctetsDropped
		oof	OutOfProfileOctetsForwarded

[17] For a SAP in AAL5 SDU mode, packet counters refer to the number of SDU. For a SAP in N-to-1 cell mode, packet counters refer to the number of cells.

Record name	Sub-record	Field	Field description
		ood	OutOfProfileOctetsDropped
Network-egress-octets (neo)	neo	port	PortId
		qid	QueueId
		iof	InProfileOctetsForwarded
		iod	InProfileOctetsDropped
		oof	OutOfProfileOctetsForwarded
		ood	OutOfProfileOctetsDropped
Network-ingress-packets (nip)	nip	port	PortId
		qid	QueueId
		ipf	InProfilePktsForwarded
		ipd	InProfilePktsDropped
		opf	OutOfProfilePktsForwarded
		opd	OutOfProfilePktsDropped
Network-egress-packets (nep)	nep	port	PortId
		qid	QueueId
		ipf	InProfilePktsForwarded
		ipd	InProfilePktsDropped
		opf	OutOfProfilePktsForwarded
		opd	OutOfProfilePktsDropped
Compact-service-ingress-octets (ctSio)	ctSio	svc	SvcId
		sap	SapId
		qid	QueueId
		hoo	OfferedHiPrioOctets
		hod	DroppedHiPrioOctets
		loo	LowOctetsOffered
		lod	LowOctetsDropped
		uco	UncoloredOctetsOffered
Combined-service-ingress (cmSipo)	cmSipo	svc	SvcId

Record name	Sub-record	Field	Field description
		sap	SapId
		qid	QueueId
		hpo	HighPktsOffered
		hpd	HighPktsDropped
		lpo	LowPktsOffered
		lpd	LowPktsDropped
		ucp	UncoloredPacketsOffered
		hoo	OfferedHiPrioOctets
		hod	DroppedHiPrioOctets
		loo	LowOctetsOffered
		lod	LowOctetsDropped
		uco	UncoloredOctetsOffered
		ipf	InProfilePktsForwarded
		opf	OutOfProfilePktsForwarded
		iof	InProfileOctetsForwarded
		oof	OutOfProfileOctetsForwarded
Combined-network-ing-egr-octets (cm Nio & cmNeo)	cmNio	port	PortId
		qid	QueueId
		iof	InProfileOctetsForwarded
		iod	InProfileOctetsDropped
		oof	OutOfProfileOctetsForwarded
		ood	OutOfProfileOctetsDropped
	cmNeo	port	PortId
		qid	QueueId
		iof	InProfileOctetsForwarded
		iod	InProfileOctetsDropped
		oof	OutOfProfileOctetsForwarded
		ood	OutOfProfileOctetsDropped

Record name	Sub-record	Field	Field description
Combined-service-ingr-egr-octets (cmSio & CmSeo)	cmSio	svc	SvcId
		sap	SapId
		qid	QueueId
		hoo	OfferedHiPrioOctets
		hod	DroppedHiPrioOctets
		loo	LowOctetsOffered
		lod	LowOctetsDropped
		uco	UncoloredOctetsOffered
		iof	InProfileOctetsForwarded
		oof	OutOfProfileOctetsForwarded
	cmSeo	svc	SvcId
		sap	SapId
		qid	QueueId
		iof	InProfileOctetsForwarded
		iod	InProfileOctetsDropped
		oof	OutOfProfileOctetsForwarded
		ood	OutOfProfileOctetsDropped
Complete-network-ingr-egr (cpNipo & cpNepo)	cpNipo	port	PortId
		qid	QueueId
		ipf	InProfilePktsForwarded
		ipd	InProfilePktsDropped
		opf	OutOfProfilePktsForwarded
		opd	OutOfProfilePktsDropped
		iof	InProfileOctetsForwarded
		iod	InProfileOctetsDropped
		oof	OutOfProfileOctetsForwarded
		ood	OutOfProfileOctetsDropped
	cpNepo	port	PortId

Record name	Sub-record	Field	Field description
		qid	QueueId
		ipf	InProfilePktsForwarded
		ipd	InProfilePktsDropped
		opf	OutOfProfilePktsForwarded
		opd	OutOfProfilePktsDropped
		iof	InProfileOctetsForwarded
		iod	InProfileOctetsDropped
		oof	OutOfProfileOctetsForwarded
		ood	OutOfProfileOctetsDropped
Complete-service-ingress-egress (cp Sipo & cpSepo)	cpSipo	svc	SvcId
		sap	SapId
		qid	QueueId
		hpo	HighPktsOffered
		hpd	HighPktsDropped
		lpo	LowPktsOffered
		lpd	LowPktsDropped
		ucp	UncoloredPacketsOffered
		hoo	OfferedHiPrioOctets
		hod	DroppedHiPrioOctets
		loo	LowOctetsOffered
		lod	LowOctetsDropped
		uco	UncoloredOctetsOffered
		apo	AllPacketsOffered
		aoo	AllOctetsOffered
		apd	AllPacketsDropped
		aod	AllOctetsDropped
		apf	AllPacketsForwarded
		aof	AllOctetsForwarded

Record name	Sub-record	Field	Field description
		ipd	InProfilePktsDropped
		iod	InProfileOctetsDropped
		opd	OutOfProfilePktsDropped
		ood	OutOfProfileOctetsDropped
		hpf	HighPriorityPacketsForwarded
		hof	HighPriorityOctetsForwarded
Complete-service-ingress-egress (cp Sipo & cpSepo) (Continued)	cpSipo (Continued)	lpf	LowPriorityPacketsForwarded
		lof	LowPriorityOctesForwarded
		ipf	InProfilePktsForwarded
		opf	OutOfProfilePktsForwarded
		iof	InProfileOctetsForwarded
		oof	OutOfProfileOctetsForwarded
	cpSepo	svc	SvcId
		sap	SapId
		qid	QueueId
		ipf	InProfilePktsForwarded
		ipd	InProfilePktsDropped
		opf	OutOfProfilePktsForwarded
		opd	OutOfProfilePktsDropped
		iof	InProfileOctetsForwarded
		iod	InProfileOctetsDropped
		oof	OutOfProfileOctetsForwarded
		ood	OutOfProfileOctetsDropped
Complete-sdp-ingress-egress (cp Sdpipo & cpSdpepo)	cpSdpipo	sdp	SdpID
		tpf	TotalPacketsForwarded
		tpd	TotalPacketsDropped
		tof	TotalOctetsForwarded
		tod	TotalOctetsDropped

Record name	Sub-record	Field	Field description
	cpSdpepo	sdp	SdpID
		tpd	TotalPacketsDropped
		tod	TotalOctetsDropped
Combined-sdp-ingress-egress (cm Sdpipo & cmSdpepo)	cmSdpipo	svc	SvcID
		sdp	SdpID
		tpf	TotalPacketsForwarded
		tpd	TotalPacketsDropped
		tof	TotalOctetsForwarded
		tod	TotalOctetsDropped
	cmSdpepo	svc	SvcID
		sdp	SdpID
		tpf	TotalPacketsForwarded
		tof	TotalOctetsForwarded
Complete-sdp-ingress-egress (cm Sdpipo & cmsdpepo) (cpSdpip & cp Sdpepo)	cmSdpipo	svc	SvcID
		sdp	SdpID
		tpf	TotalPacketsForwarded
		tpd	TotalPacketsDropped
		tof	TotalOctetsForwarded
		tod	TotalOctetsDropped
	cmSdpepo	svc	SvcID
		sdp	SdpID
		tpf	TotalPacketsForwarded
		tof	TotalOctetsForwarded
	cpSdpipo	sdp	SdpID
		tpf	TotalPacketsForwarded
		tpd	TotalPacketsDropped
		tof	TotalOctetsForwarded
		tod	TotalOctetsDropped

Record name	Sub-record	Field	Field description
	cpSdpepo	sdp	SdpID
		tpf	TotalPacketsForwarded
		tof	TotalOctetsForwarded
Complete-subscriber-ingress-egress (cpSBipo & cpSBepo)	Subscriber Information	subId	SubscriberId
		subProfile	SubscriberProfile
	Sla- Information	svc	SvcId
		sap	SapId
		slaProfile	SlaProfile
		spiSharing	SPI sharing type and identifier
Complete-subscriber-ingress-egress (cpSBipo & cpSBepo) (Continued)	cpSBipo	qid	QueueId
		hpo	HighPktsOffered
		hpd	HighPktsDropped
		lpo	LowPktsOffered
		lpd	LowPktsDropped
		ucp	UncolouredPacketsOffered
		hoo	OfferedHiPrioOctets
		hod	DroppedHiPrioOctets
		loo	LowOctetsOffered
		lod	LowOctetsDropped
		apo	AllPktsOffered
		aoo	AllOctetsOffered
		uco	UncolouredOctetsOffered
		ipf	InProfilePktsForwarded
		opf	OutOfProfilePktsForwarded
		iof	InProfileOctetsForwarded
		oof	OutOfProfileOctetsForwarded
		v4pf	IPv4PktsForwarded
		v6pf	IPv6PktsForwarded

Record name	Sub-record	Field	Field description
		v4pd	IPv4PktsDropped
		v6pd	IPv6PktsDropped
		v4of	IPv4OctetsForwarded
		v6of	IPv6OctetsForwarded
		v4od	IPv4OctetsDropped
		v6od	IPv6OctetsDropped
Complete-subscriber-ingress-egress (cpSBipo & cpSBepo) (Continued)	cpSBepo	qid	QueueId
		ipf	InProfilePktsForwarded
		ipd	InProfilePktsDropped
		opf	OutOfProfilePktsForwarded
		opd	OutOfProfilePktsDropped
		iof	InProfileOctetsForwarded
		iod	InProfileOctetsDropped
		oof	OutOfProfileOctetsForwarded
		ood	OutOfProfileOctetsDropped
		v4pf	IPv4PktsForwarded
		v6pf	IPv6PktsForwarded
		v4pd	IPv4PktsDropped
		v6pd	IPv6PktsDropped
		v4of	IPv4OctetsForwarded
		v6of	IPv6OctetsForwarded
		v4od	IPv4OctetsDropped
		v6od	IPv6OctetsDropped
saa	saa	tmd	TestMode
		own	OwnerName
		tst	TestName
		png	PingRun subrecord
		rid	RunIndex

Record name	Sub-record	Field	Field description
		trr	TestRunResult
		mnr	MinRtt
		mxr	MaxRtt
		avr	AverageRtt
		rss	RttSumOfSquares
		pbr	ProbeResponses
		spb	SentProbes
		mnt	MinOutTt
		mxt	MaxOutTt
		avt	AverageOutTt
		tss	OutTtSumOfSquares
		mni	MinInTt
		mxi	MaxInTt
		avi	AverageInTt
		iss	InTtSumOfSqrs
		ojt	OutJitter
		ijt	InJitter
		rjt	RtJitter
		prt	ProbeTimeouts
		prf	ProbeFailures
saa (Continued)	trc	rid	RunIndex
		trr	TestRunResult
		lgp	LastGoodProbe
	hop	hop	TraceHop
		hid	HopIndex
		mnr	MinRtt
		mxr	MaxRtt
		avr	AverageRtt

Record name	Sub-record	Field	Field description
		rss	RttSumOfSquares
		pbr	ProbeResponses
		spb	SentProbes
		mnt	MinOutTt
		mxt	MaxOutTt
		avt	AverageOutTt
		tss	OutTtSumOfSquares
		mni	MinInTt
		mxi	MaxInTt
		avi	AverageInTt
		iss	InTtSumOfSqrs
		ojt	OutJitter
		ijt	InJitter
		rjt	RtJitter
		prt	ProbeTimeouts
		prf	ProbeFailures
		tat	TraceAddressType
		tav	TraceAddressValue
Complete-ethernet-port (enet)	enet	port	PortId
		to	EtherStatsOctets
		tp	EtherStatsPkts
		de	EtherStatsDropEvents
		tbcp	EtherStatsBroadcastPkts
		mcp	EtherStatsMulticastPkts
		cae	EtherStatsCRCAlignErrors
		up	EtherStatsUndersizePkts
		op	EtherStatsOversizePkts
		fgm	EtherStatsFragments

Record name	Sub-record	Field	Field description
		jab	EtherStatsJabbers
		col	EtherStatsCollisions
		p64o	EtherStatsPkts64Octets
		p127o	EtherStatsPkts65to127Octets
		p255o	EtherStatsPkts128to255Octets
		p511o	EtherStatsPkts256to511Octets
		p1023o	EtherStatsPkts512to1023Octets
		p1518o	EtherStatsPkts1024to1518Octets
		po1518o	EtherStatsPktsOver1518Octets
		ae	Dot3StatsAlignmentErrors
		fe	Dot3StatsFCSErrors
		scf	Dot3StatsSingleCollisionFrames
		mcf	Dot3StatsMultipleCollisionFrames
		sqe	Dot3StatsSQETestErrors
		dt	Dot3StatsDeferredTransmissions
Complete-ethernet-port (enet) (Continued)	enet (Continued)	lcc	Dot3StatsLateCollisions
		exc	Dot3StatsExcessiveCollisions
		imt	Dot3StatsInternalMacTransmit Errors
		cse	Dot3StatsCarrierSenseErrors
		ftl	Dot3StatsFrameTooLongs
		imre	Dot3StatsInternalMacReceive Errors
		se	Dot3StatsSymbolErrors
		ipf	Dot3InPauseFrames
		opf	Dot3OutPauseFrames

Table 47: Policer stats field descriptions, Table 48: Queue group record types, and Table 49: Queue group record type fields provide field descriptions.

The actual fields present in policer stats accounting records depend on the configured **stat-mode** of the policer associated with the record.

Table 47: Policer stats field descriptions

Field	Field description
pid	PolicerId
statmode	PolicerStatMode
aod	AllOctetsDropped
aof	AllOctetsForwarded
aoo	AllOctetsOffered
apd	AllPacketsDropped
apf	AllPacketsForwarded
apo	AllPacketsOffered
c1od	ConnectionOneOctetsDropped[18]
c1of	ConnectionOneOctetsForwarded[18]
c1oo	ConnectionOneOctetsOffered[18]
c1pd	ConnectionOnePacketsDropped[18]
c1pf	ConnectionOnePacketsForwarded[18]
c1po	ConnectionOnePacketsOffered[18]
c2od	ConnectionTwoOctetsDropped[18]
c2of	ConnectionTwoOctetsForwarded[18]
c2oo	ConnectionTwoOctetsOffered[18]
c2pd	ConnectionTwoPacketsDropped[18]
c2pf	ConnectionTwoPacketsForwarded[18]
c2po	ConnectionTwoPacketsOffered[18]
hod	HighPriorityOctetsDropped
hof	HighPriorityOctetsForwarded

[18] Enhanced Subscriber Management (ESM) connection bonding only

Field	Field description
hoo	HighPriorityOctetsOffered
hpd	HighPriorityPacketsDropped
hpf	HighPriorityPacketsForwarded
hpo	HighPriorityPacketsOffered
iod	InProfileOctetsDropped
iof	InProfileOctetsForwarded
ioo	InProfileOctetsOffered
ipd	InProfilePacketsDropped
ipf	InProfilePacketsForwarded
ipo	InProfilePacketsOffered
lod	LowPriorityOctetsDropped
lof	LowPriorityOctetsForwarded
loo	LowPriorityOctetsOffered
lpd	LowPriorityPacketsDropped
lpf	LowPriorityPacketsForwarded
lpo	LowPriorityPacketsOffered
opd	OutOfProfilePacketsDropped
opf	OutOfProfilePacketsForwarded
opo	OutOfProfilePacketsOffered
ood	OutOfProfileOctetsDropped
oof	OutOfProfileOctetsForwarded
ooo	OutOfProfileOctetsOffered
xpd	ExceedProfilePktsDropped
xpf	ExceedProfilePktsForwarded
xpo	ExceedProfilePktsOffered
xod	ExceedProfileOctetsDropped
xof	ExceedProfileOctetsForwarded
xoo	ExceedProfileOctetsOffered

Field	Field description
ppd	InplusProfilePacketsDropped
ppf	InplusProfilePacketsForwarded
ppo	InplusProfilePacketsOffered
pod	InplusProfileOctetsDropped
pof	InplusProfileOctetsForwarded
poo	InplusProfileOctetsOffered
uco	UncoloredOctetsOffered
ucp	UncoloredPacketsOffered
v4po	IPv4PktsOffered [19]
v4oo	IPv4OctetsOffered[19]
v6po	IPv6PktsOffered[19]
v6oo	IPv6OctetsOffered[19]
v4pf	IPv4PktsForwarded[19]
v6pf	IPv6PktsForwarded[19]
v4pd	IPv4PktsDropped[19]
v6pd	IPv6PktsDropped[19]
v4of	IPv4OctetsForwarded[19]
v6of	IPv6OctetsForwarded[19]
v4od	IPv4OctetsDropped[19]
v6od	IPv6OctetsDropped[19]

Table 48: Queue group record types

Record name	Description
qgone	PortQueueGroupOctetsNetworkEgress
qgosi	PortQueueGroupOctetsServiceIngress
qgose	PortQueueGroupOctetsServiceEgress

[19] Enhanced Subscriber Management (ESM) only

Record name	Description
qgpne	PortQueueGroupPacketsNetworkEgress
qgpsi	PortQueueGroupPacketsServiceIngress
qgpse	PortQueueGroupPacketsServiceEgress
fpqgosi	ForwardingPlaneQueueGroupOctetsServiceIngress
fpqgoni	ForwardingPlaneQueueGroupOctetsNetworkIngress
fpqgpsi	ForwardingPlaneQueueGroupPacketsServiceIngress
fpqgpni	ForwardingPlaneQueueGroupPacketsNetworkIngress

Table 49: Queue group record type fields

Field	Field description
data port	Port (used for port based Queue Groups)
member-port	LAGMemberPort (used for port based Queue Groups)
data slot	Slot (used for Forwarding Plane based Queue Groups)
forwarding-plane	ForwardingPlane (used for Forwarding Plane based Queue Groups)
queue-group	QueueGroupName
instance	QueueGroupInstance
qid	QueueId
pid	PolicerId
statmode	PolicerStatMode
aod...ucp	same as above

7.7.2 Accounting files

When a policy is created and applied to a service or network port, the accounting file is stored on the compact flash in a compressed XML file format. The router creates two directories on the compact flash to store the files.

Example

The following output displays a directory named `\act-collect` that holds open accounting files that are actively collecting statistics. The directory named `\act` stores the files that have been closed and are awaiting retrieval.

```
A:node-2>file dir cf1:\act*
```

```
12/19/2006 06:08a <DIR> act-collect
12/19/2006 06:08a <DIR> act

A:node-2>file dir cf1:\act-collect\
Directory of cf1:\act-collect#
12/23/2006 01:46a <DIR> .
12/23/2006 12:47a <DIR>  ..
12/23/2006 01:46a 112 act1111-20031223-014658.xml.gz
12/23/2006 01:38a 197 act1212-20031223-013800.xml.gz
```

Accounting files always have the prefix "act" followed by the accounting policy ID, log ID, and timestamp. For detailed information about the accounting log file naming and log file policy properties such as rollover and retention, see Log and accounting files.

7.7.3 Design considerations for accounting policies

The router has ample resources to support large scale accounting policy deployments. When preparing for an accounting policy deployment, verify that data collection, file rollover, and file retention intervals are properly tuned for the amount of statistics to be collected.

If the accounting policy collection interval is too brief there may be insufficient time to store the data from all the services within the specified interval. If that is the case, some records may be lost or incomplete. Interval time, record types, and number of services using an accounting policy are all factors that should be considered when implementing accounting policies.

The rollover and retention intervals on the log files and the frequency of file retrieval must also be considered when designing accounting policy deployments. The amount of data stored depends on the type of record collected, the number of services that are collecting statistics, and the collection interval that is used. For example, with a 1Gb CF and using the default collection interval, the system is expected to hold 48 hours' worth of billing information.

7.7.4 Reporting and time-based accounting

SR OS on the 7705 SAR Gen 2 supports volume accounting and time-based accounting concepts, and provides an extra level of intelligence at the network element level to provide service models such as "prepaid access" in a scalable manner. This means that the network element gathers and stores per-subscriber accounting information and compares it with "pre-defined" quotas. When a quota is exceeded, the pre-defined action (such as re-direction to a web portal or disconnect) is applied.

7.7.5 Custom record usage for overhead reduction in accounting

Custom records can be used to decrease accounting messaging overhead as follows:

- User configurable records
- Changed statistics only
- Configurable accounting records
- Significant change only reporting

7.7.5.1 User configurable records

Users can define a collection of fields that make up a record. These records can be assigned to an accounting policy. These are user-defined records instead of being limited to pre-defined record types. The operator can select queues and policers and the counters within these queues and policers that need to be collected. See the predefined records containing a specific field for XML field name of a custom record field.

7.7.5.2 Changed statistics only

A record is only generated if a significant change has occurred to the fields being written in a specific record. This capability applies to both ingress and egress records regardless on the method of delivery (such as RADIUS and XML). The capability also applies to Application Assurance records; however without an ability to specify different significant change values and per-field scope (for example, all fields of a custom record are collected if any activity was reported against any of the statistics that are part of the custom record).

7.7.5.3 Configurable accounting records

7.7.5.3.1 XML accounting files for service

To reduce the volume of data generated, you can specify which records are needed for collection. This excludes queues and policers or selected counters within the queues and policers that are not relevant for billing.

Use the commands in the following context to configure custom records.

```
configure log accounting-policy custom-record
```

Record headers including information such as service ID or SAP ID are always generated.

7.7.5.3.2 XML accounting files for policer counters

Policer counters can be collected using custom records within the accounting policy configuration. The policer identifier for which counters are collected must be configured under **custom-record**, specifying the required ingress (**i-counters**) and egress (**e-counters**) counters to be collected. A similar configuration is available for a reference policer (**ref-policer**) to define a reference counter used together with the **significant-change** command.

The counters collected are dependent on the **stat-mode** of the related policer, as this determines which statistics are collected by the system for the policer.

The ingress policer counters collected for each combination of XML accounting record name and policer **stat-mode** are provided in Table 50: Custom record policer ingress counter mapping.

The egress policer counters collected for each combination of XML accounting record name and policer **stat-mode** are provided in Table 51: Custom record policer egress counter mapping.

Table 50: Custom record policer ingress counter mapping

Policer i-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
in-profile-octets-discarded-count	minimal	—	—
	offered-limited-capped-cir offered-limited-profile-cir offered-priority-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir	In-Profile Octets Dropped	iod
	offered-priority-no-cir	High-Priority Octets Dropped	hod
	v4-v6	V4 Octets Dropped	v4od
in-profile-octets-forwarded-count	minimal	—	—
	offered-limited-capped-cir offered-limited-profile-cir offered-priority-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir	In-Profile Octets Forwarded	iof
	offered-priority-no-cir	High-Priority Octets Forwarded	hof
	v4-v6	V4 Octets Forwarded	v4of
in-profile-octets-offered-count	minimal offered-limited-profile-cir offered-total-cir	—	—
	offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir	In-Profile Octets Offered	ioo
	offered-priority-cir offered-priority-no-cir	High-Priority Octets Offered	hoo

Policer i-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	v4-v6	V4 Octets Offered	v4oo
in-profile-packets-discarded-count	minimal	—	—
	offered-limited-capped-cir offered-limited-profile-cir offered-priority-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir	In-Profile Packets Dropped	ipd
	offered-priority-no-cir	High-Priority Packets Dropped	hpd
	v4-v6	V4 Packets Dropped	v4pd
in-profile-packets-forwarded-count	minimal	—	—
	offered-limited-capped-cir offered-limited-profile-cir offered-priority-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir	In-Profile Packets Forwarded	ipf
	offered-priority-no-cir	High-Priority Packets Forwarded	hpf
	v4-v6	V4 Packets Forwarded	v4pf
in-profile-packets-offered-count	minimal offered-limited-profile-cir offered-total-cir	—	—
	offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir	In-Profile Packets Offered	ipo
	offered-priority-cir offered-priority-no-cir	High-Priority Packets Offered	hpo

Policer i-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	v4-v6	V4 Packets Offered	v4po
out-profile-octets-discarded-count	minimal	All Octets Dropped	aod
	offered-limited-capped-cir offered-limited-profile-cir offered-priority-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir	Out-of-Profile Octets Dropped	ood
	offered-priority-no-cir	Low-Priority Octets Dropped	lod
	v4-v6	V6 Octets Dropped	v6od
out-profile-octets-forwarded-count	minimal	All Octets Forwarded	aof
	offered-limited-capped-cir offered-limited-profile-cir offered-priority-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir	Out-of-Profile Octets Forwarded	oof
	offered-priority-no-cir	Low-Priority Octets Forwarded	lof
	v4-v6	V6 Octets Forwarded	v6of
out-profile-octets-offered-count	minimal offered-total-cir	All Octets Offered	aoo
	offered-limited-capped-cir	—	—
	offered-limited-profile-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir	Out-of-Profile Octets Offered	ooo
	offered-priority-cir offered-priority-no-cir	Low-Priority Octets Offered	loo

Policer i-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	v4-v6	V6 Octets Offered	v6oo
out-profile-packets-discarded-count	minimal	All Packets Dropped	apd
	offered-limited-capped-cir offered-limited-profile-cir offered-priority-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir	Out-of-Profile Packets Dropped	opd
	offered-priority-no-cir	Low-Priority Packets Dropped	lpd
	v4-v6	V6 Packets Dropped	v6pd
out-profile-packets-forwarded-count	minimal	All Packets Forwarded	apf
	offered-limited-capped-cir offered-limited-profile-cir offered-priority-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir	Out-of-Profile Packets Forwarded	opf
	offered-priority-no-cir	Low-Priority Packets Forwarded	lpf
	v4-v6	V6 Packets Forwarded	v6pf
out-profile-packets-offered-count	minimal offered-total-cir	All Packets Offered	apo
	offered-limited-capped-cir	n/a	n/a
	offered-limited-profile-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir	Out-of-Profile Packets Offered	opo
	offered-priority-cir offered-priority-no-cir	Low-Priority Packets Offered	lpo

Policer i-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	v4-v6	V6 Packets Offered	v6po
uncoloured-octets-offered-count	minimal offered-priority-cir offered-priority-no-cir offered-profile-no-cir offered-total-cir v4-v6	—	—
	offered-limited-capped-cir offered-limited-profile-cir offered-profile-capped-cir offered-profile-cir	Uncoloured Octets Offered	uco
uncoloured-packets-offered-count	minimal offered-priority-cir offered-priority-no-cir offered-profile-no-cir offered-total-cir v4-v6	—	—
	offered-limited-capped-cir offered-limited-profile-cir offered-profile-capped-cir offered-profile-cir	Uncoloured Packets Offered	ucp

Table 51: Custom record policer egress counter mapping

Policer e-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
exceed-profile-octets-discarded-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir	n/a	n/a

Policer e-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	v4-v6		
	offered-four-profile-no-cir offered-total-cir-exceed offered-total-cir-four-profile	Exceed-Profile Octets Dropped	xod
exceed-profile-octets-forwarded-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir v4-v6	—	—
	offered-four-profile-no-cir offered-total-cir-exceed offered-total-cir-four-profile	Exceed-Profile Octets Forwarded	xof
exceed-profile-octets-offered-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile v4-v6	—	—
	offered-four-profile-no-cir	Exceed-Profile Octets Offered	xoo
exceed-profile-packets-discarded-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir	—	—

Policer e-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	offered-total-cir v4-v6		
	offered-four-profile-no-cir offered-total-cir-exceed offered-total-cir-four-profile	Exceed-Profile Packets Dropped	xpd
exceed-profile-packets-forwarded-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir v4-v6	—	—
	offered-four-profile-no-cir offered-total-cir-exceed offered-total-cir-four-profile	Exceed-Profile Packets Forwarded	xpf
exceed-profile-packets-offered-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile v4-v6	—	—
	offered-four-profile-no-cir	Exceed-Profile Packets Offered	xpo
in-plus-profile-octets-discarded-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir	—	—

Policer e-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	offered-profile-no-cir offered-total-cir offered-total-cir-exceed v4-v6		
	offered-four-profile-no-cir offered-total-cir-four-profile	In-Plus-Profile Octets Dropped	pod
in-plus-profile-octets-forwarded-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed v4-v6	—	—
	offered-four-profile-no-cir offered-total-cir-four-profile	In-Plus-Profile Octets Forwarded	pof
in-plus-profile-octets-offered-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile v4-v6	—	—
	offered-four-profile-no-cir	In-Plus-Profile Octets Offered	poo
in-plus-profile-packets-discarded-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir	—	—

Policer e-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed v4-v6		
	offered-four-profile-no-cir offered-total-cir-four-profile	In-Plus-Profile Packets Dropped	ppd
in-plus-profile-packets-forwarded-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed v4-v6	—	—
	offered-four-profile-no-cir offered-total-cir-four-profile	In-Plus-Profile Packets Forwarded	ppf
in-plus-profile-packets-offered-count	bonding minimal offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile v4-v6	—	—
	offered-four-profile-no-cir	In-Plus-Profile Packets Offered	ppo
in-profile-octets-discarded-count	bonding	Connection 1 Octets Dropped	c1od
	minimal	—	—

Policer e-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	offered-four-profile-no-cir offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile	In-Profile Octets Dropped	iod
	v4-v6	V4 Octets Dropped	v4od
in-profile-octets-forwarded-count	bonding	Connection 1 Octets Forwarded	c1of
	minimal	—	—
	offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-four-profile-no-cir offered-total-cir-four-profile	In-Profile Octets Forwarded	iof
	v4-v6	V4 Octets Forwarded	v4of
in-profile-octets-offered-count	bonding	Connection 1 Octets Offered	c1oo
	minimal offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile	—	—
	offered-four-profile-no-cir offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir	In-Profile Octets Offered	ioo
	v4-v6	V4 Octets Offered	v4oo

Policer e-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
in-profile-packets-discarded-count	bonding	Connection 1 Packets Dropped	c1pd
	minimal	—	—
	offered-four-profile-no-cir offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile	In-Profile Packets Dropped	ipd
	v4-v6	V4 Packets Dropped	v4pd
in-profile-packets-forwarded-count	bonding	Connection 1 Packets Forwarded	c1pf
	minimal	—	—
	offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-four-profile-no-cir offered-total-cir-four-profile	In-Profile Packets Forwarded	ipf
	v4-v6	V4 Packets Forwarded	v4pf
in-profile-packets-offered-count	bonding	Connection 1 Packets Offered	c1po
	minimal offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile	—	—
	offered-four-profile-no-cir offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir	In-Profile Packets Offered	ipo

Policer e-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	offered-profile-no-cir		
	v4-v6	V4 Packets Offered	v4po
out-profile-octets-discarded-count	bonding	Connection 2 Octets Dropped	c2od
	minimal	All Octets Dropped	aod
	offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-four-profile-no-cir offered-total-cir-four-profile	Out-of-Profile Octets Dropped	ood
	v4-v6	V6 Octets Dropped	v6od
out-profile-octets-forwarded-count	bonding	Connection 2 Octets Forwarded	c2of
	minimal	All Octets Forwarded	aof
	offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-four-profile-no-cir offered-total-cir-four-profile	Out-of-Profile Octets Forwarded	oof
	v4-v6	V6 Octets Forwarded	v6of
out-profile-octets-offered-count	bonding	Connection 2 Octets Offered	c2oo
	minimal offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile	All Octets Offered	aoo
	offered-limited-capped-cir offered-profile-capped-cir	Out-of-Profile Octets Offered	ooo

Policer e-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	offered-profile-cir offered-profile-no-cir offered-four-profile-no-cir		
	v4-v6	V6 Octets Offered	v6oo
out-profile-packets-discarded-count	bonding	Connection 2 Packets Dropped	c2pd
	minimal	All Packets Dropped	apd
	offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-four-profile-no-cir offered-total-cir-four-profile	Out-of-Profile Packets Dropped	opd
	v4-v6	V6 Packets Dropped	v6pd
out-profile-packets-forwarded-count	bonding	Connection 2 Packets Forwarded	c2pf
	minimal	All Packets Forwarded	apf
	offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-four-profile-no-cir offered-total-cir-four-profile	Out-of-Profile Packets Forwarded	opf
	v4-v6	V6 Packets Forwarded	v6pf
out-profile-packets-offered-count	bonding	Connection 2 Packets Offered	c2po
	minimal offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile	All Packets Offered	apo

Policer e-counters CLI name	Policer stat-mode	Custom record counter	Custom record field
	offered-limited-capped-cir offered-profile-capped-cir offered-profile-cir offered-profile-no-cir offered-four-profile-no-cir	Out-of-Profile Packets Offered	opo
	v4-v6	V6 Packets Offered	v6po
uncoloured-octets-offered-count	bonding minimal offered-four-profile-no-cir offered-limited-capped-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile v4-v6	—	—
	offered-profile-capped-cir offered-profile-cir	Uncoloured Octets Offered	uco
uncoloured-packets-offered-count	bonding minimal offered-four-profile-no-cir offered-limited-capped-cir offered-profile-no-cir offered-total-cir offered-total-cir-exceed offered-total-cir-four-profile v4-v6	—	—
	offered-profile-capped-cir offered-profile-cir	Uncoloured Packets Offered	ucp

7.7.5.3.3 RADIUS accounting in networks using ESM

You can include individual counters in RADIUS accounting messages. Use the commands in the following context to configure custom-record counters for RADIUS accounting messages.

```
configure subscriber-mgmt radius-accounting-policy custom-record
```

7.7.5.4 Significant change only reporting

Another way to decrease accounting messaging related to overhead is to include only "active" objects in a periodical reporting. An "active object" in this context is an object which has seen a "significant" change in corresponding counters. A significant change is defined in terms of a cumulative value (the sum of all reference counters).

This concept is applicable to all methods used for gathering accounting information, such as an XML file and RADIUS, as well as to all applications using accounting, such as service-acct, ESM-acct, and Application Assurance.

Accounting records are reported at the periodical intervals. This periodic reporting is extended with an internal filter which omits periodical updates for objects whose counter change experienced lower changes than a defined (configurable) threshold.

Specific to RADIUS accounting the **significant-change** command does not affect ACCT-STOP messages. ACCT-STOP messages are always sent, regardless the amount of change of the corresponding host.

For Application Assurance records, a significant change of 1 in any field of a customized record (send a record if any field changed) is supported. When configured, if any statistic field records activity, an accounting record containing all fields is collected.

7.7.6 Immediate completion of records

7.7.6.1 Record completion for XML accounting

For ESM RADIUS accounting, an accounting stop message is sent when:

- A subscriber/subscriber-host is deleted.
- An SLA profile instance is changed.

A similar concept is also used for XML accounting. In case the accounted object is deleted or changed, the latest information is written in the XML file with a "final" tag indication in the record header.

7.7.7 AA accounting per forwarding class

This feature allows the operator to report on protocol/application/app-group volume usage per forwarding class by adding a bitmap information representing the observed FC in the XML accounting files. In case the accounted object is deleted or changed, the latest information is written in the XML file with a "final" tag indication in the record header.

7.8 Configuration notes

This section describes logging configuration restrictions.

- A log file policy or log filter policy cannot be deleted if it has been applied to a log.
- File policies, syslog policies, or SNMP trap groups must be configured before they can be applied to a log ID.
- A file policy can only be assigned to either one event log or one accounting policy.
- Accounting policies must be configured in the **configure log** context before they can be applied to a service SAP or service interface, or applied to a network port.
- The SNMP trap ID must be the same as the log ID.

7.9 Configuring logging with CLI

This section provides information to configure logging using the command line interface.

7.9.1 Log configuration overview

Configure logging to save information in a log file or direct the messages to other devices. Logging does the following:

- Provides you with logging information for monitoring and troubleshooting.
- Allows the selection of the types of logging information to be recorded.
- Allows the assignment of a severity to the log messages.
- Allows the selection of source and target of logging information.

7.9.2 Log types

Logs can be configured in the following contexts:

- **log file**

  Log files can contain log event message streams or accounting/billing information. Log file policies are used to direct events, alarms, traps, and debug information to a file on local storage devices (for example, cf2:).

- **SNMP trap groups**

  SNMP trap groups contain an IP address and community names which identify targets to send traps following specified events.

- **syslog**

  Information can be sent to a syslog host that is capable of receiving selected Syslog messages from a network element.

- **event control**

Configures a particular event or all events associated with an application to be generated or suppressed.

- **event filters**

  An event filter defines whether to forward or drop an event or trap based on match criteria.

- **accounting policies**

  An accounting policy defines the accounting records that will be created. Accounting policies can be applied to one or more service access points (SAPs).

- **event logs**

  An event log defines the types of events to be delivered to its associated destination.

- **event throttling rate**

  Defines the rate of throttling events.

7.9.3 Basic log configuration

The most basic log configuration must have the following:

- log ID or accounting policy ID
- log source
- log destination

The following example displays a log configuration.

Example: MD-CLI

```
[ex:/configure log]
A:admin@node-2# info
    log-events {
        bgp event sendNotification {
            severity critical
            throttle false
        }
    }
    file "1" {
        description "This is a test file-id."
        compact-flash-location {
            primary cf1
        }
    }
    file "2" {
        description "This is a test log."
        compact-flash-location {
            primary cf1
        }
    }
    log-id "2" {
        source {
            main true
        }
        destination {
            file "2"
        }
    }
    snmp-trap-group "7" {
```

```
        trap-target "testTarget" {
            address 11.22.33.44
            version snmpv2c
            notify-community "public"
        }
    }
```

Example: classic CLI

```
A:node-2>config>log# info
#------------------------------------------
echo "Log Configuration "
#------------------------------------------
        event-control 2005 generate critical
        file-id 1
            description "This is a test file-id."
            location cf1:
        exit
        file-id 2
            description "This is a test log."
            location cf1:
        exit
        snmp-trap-group 7
            trap-target 11.22.33.44 "snmpv2c" notify-community "public"
        exit
        log-id 2
            from main
            to file 2
        exit
----------------------------------------------
```

7.9.4 Common configuration tasks

The following sections describe basic system tasks that must be performed.

7.9.4.1 Configuring an event log

A log file policy contains information used to direct events, alarms, traps, and debug information to a file on a local storage device (for example, cf2:). One or more event sources can be specified. File policies, SNMP trap groups, or syslog policies must be configured before they can be applied to an event log.

Use commands in the following context to configure an event log file.

```
configure log log-id
```

The following example shows an event log file configuration.

Example: MD-CLI

```
[ex:/configure log]
A:admin@node-2# info
    log-id "2" {
        description "This is a test log file."
        filter "1"
        source {
            main true
            security true
```

```
        }
        destination {
            file "1"
        }
    }
```

Example: classic CLI

```
A:node-2>config>log>log-id$ info
----------------------------------------------
...
log-id 2 name "2"
    description "This is a test log file."
    filter 1
    from main security
    to file 1
exit
...
----------------------------------------------
```

7.9.4.2 Configuring a log file policy

To create a log file, a file policy is defined, the target CF or USB drive is specified, and the rollover and retention interval for the log file is defined. The rollover interval is defined in minutes and determines how long a file is used before it is closed and a new log file is created. The retention interval determines how long the file is stored on the storage device before it is deleted.

When creating new log files in a compact flash disk card, the minimum amount of free space is the minimum of 10% of Compact Flash disk capacity or 5 Mb (5,242,880 = 5 × 1024 × 1024).

The following example shows a log file configuration.

Example: MD-CLI

```
[ex:/configure log]
A:admin@node-2# info
    file "1" {
        description "This is a log file."
        rollover 600
        retention 24
        compact-flash-location {
            primary cf1
        }
    }
```

Example: classic CLI

```
A:node-2>config>log# info
----------------------------------------
        file-id 1 name "1"
            description "This is a log file."
            location cf1:
            rollover 600 retention 24
        exit
----------------------------------------------
```

7.9.4.3 Configuring an accounting policy

A log file policy must be created to collect the accounting records. The files are stored in system memory of compact flash (cf1: or cf2:) in a compressed (tar) XML format and can be retrieved using FTP or SCP. See Configuring an event log and Configuring a log file policy.

Accounting policies must be configured in the **configure log** context before they can be applied to a service SAP or service interface, or applied to a network port.

The default accounting policy statement cannot be applied to LDP nor RSVP statistics collection records.

An accounting policy must define a record type and collection interval. Only one record type can be configured per accounting policy.

When creating accounting policies, one service accounting policy and one network accounting policy can be defined as default. If statistics collection is enabled on a SAP or network port and no accounting policy is applied, then the respective default policy is used. If no default policy is defined, then no statistics are collected unless a specifically defined accounting policy is applied.

The following example shows an accounting policy configuration.

Example: MD-CLI

```
[ex:/configure log]
A:admin@node-2# info
    accounting-policy 4 {
        description "This is the default accounting policy."
        default true
        record complete-service-ingress-egress
        destination {
            file "1"
        }
    }
    accounting-policy 5 {
        description "This is a test accounting policy."
        record service-ingress-packets
        destination {
            file "3"
        }
    }
```

Example: classic CLI

```
A:node-2>config>log# info
----------------------------------------------
accounting-policy 4
    description "This is the default accounting policy."
    record complete-service-ingress-egress
    default
    to file 1
exit
accounting-policy 5
    description "This is a test accounting policy."
    record service-ingress-packets
    to file 3
exit
----------------------------------------------
```

7.9.4.4 Configuring an accounting custom record

The following example shows a custom-record configuration.

Example: Custom-record configuration (MD-CLI)

```
ex:/configure log accounting-policy 1]
A:admin@node-2# info
    custom-record {
        significant-change 20
        queue 1 {
            e-counters {
                in-profile-octets-discarded-count true
                in-profile-octets-forwarded-count true
                out-profile-octets-discarded-count true
                out-profile-octets-forwarded-count true
            }
            i-counters {
                high-octets-discarded-count true
                in-profile-octets-forwarded-count true
                low-octets-discarded-count true
                out-profile-octets-forwarded-count true
            }
        }
        ref-queue {
            all
            e-counters {
                in-profile-packets-forwarded-count true
                out-profile-packets-forwarded-count true
            }
            i-counters {
                in-profile-packets-forwarded-count true
                out-profile-packets-forwarded-count true
            }
        }
    }
```

Example: Custom-record configuration (MD-CLI)

```
[ex:/configure log accounting-policy 1]
A:admin@node-2# info
    custom-record {
        significant-change 1
        aa-specific {
            aa-sub-counters {
                long-duration-flow-count true
                medium-duration-flow-count true
                short-duration-flow-count true
                total-flow-duration true
                total-flows-completed-count true
            }
            from-aa-sub-counters {
                flows-active-count true
                flows-admitted-count true
                flows-denied-count true
                forwarding-class true
                max-throughput-octet-count true
                max-throughput-packet-count true
                max-throughput-timestamp true
                octets-admitted-count true
                octets-denied-count true
```

```
                packets-admitted-count true
                packets-denied-count true
            }
            to-aa-sub-counters {
                flows-active-count true
                flows-admitted-count true
                flows-denied-count true
                forwarding-class true
                max-throughput-octet-count true
                max-throughput-packet-count true
                max-throughput-timestamp true
                octets-admitted-count true
                octets-denied-count true
                packets-admitted-count true
                packets-denied-count true
            }
        }
        ref-aa-specific-counter {
            any true
        }
    }
```

Example: Custom-record configuration (classic CLI)

```
A:node-2>config>log>acct-plcy# info
----------------------------------------------
...
            custom-record
                queue 1
                    i-counters
                        high-octets-discarded-count
                        low-octets-discarded-count
                        in-profile-octets-forwarded-count
                        out-profile-octets-forwarded-count
                    exit
                    e-counters
                        in-profile-octets-forwarded-count
                        in-profile-octets-discarded-count
                        out-profile-octets-forwarded-count
                        out-profile-octets-discarded-count
                    exit
                exit
                significant-change 20
                ref-queue all
                    i-counters
                        in-profile-packets-forwarded-count
                        out-profile-packets-forwarded-count
                    exit
                    e-counters
                        in-profile-packets-forwarded-count
                        out-profile-packets-forwarded-count
                    exit
                exit
...
----------------------------------------------
```

Example: Custom-record configuration (classic CLI)

```
A:node-2>config>log>acct-policy# info
----------------------------------------------
...
                custom-record
```

```
                    aa-specific
                        aa-sub-counters
                            short-duration-flow-count
                            medium-duration-flow-count
                            long-duration-flow-count
                            total-flow-duration
                            total-flows-completed-count
                        exit
                        from-aa-sub-counters
                            flows-admitted-count
                            flows-denied-count
                            flows-active-count
                            packets-admitted-count
                            octets-admitted-count
                            packets-denied-count
                            octets-denied-count
                            max-throughput-octet-count
                            max-throughput-packet-count
                            max-throughput-timestamp
                            forwarding-class
                        exit
                        to-aa-sub-counters
                            flows-admitted-count
                            flows-denied-count
                            flows-active-count
                            packets-admitted-count
                            octets-admitted-count
                            packets-denied-count
                            octets-denied-count
                            max-throughput-octet-count
                            max-throughput-packet-count
                            max-throughput-timestamp
                            forwarding-class
                        exit
                    exit
                    significant-change 1
                    ref-aa-specific-counter any
...
-------------------------------------------------
```

7.9.4.5 Configuring event control

The following example shows an event control configuration.

Example: MD-CLI

```
[ex:/configure log]
A:admin@node-2# info
    log-events {
        ospf event tmnxOspfVirtIfStateChange {
            generate false
            throttle false
        }
        ospf event tmnxOspfVirtNbrStateChange {
            severity cleared
            throttle false
        }
        ospf event tmnxOspfLsdbOverflow {
            severity critical
            throttle false
        }
```

```
    }
    throttle-rate {
        limit 500
        interval 10
    }
```

Example: classic CLI

```
A:node-2>config>log# info
#------------------------------------------
echo "Log Configuration"
#------------------------------------------
        throttle-rate 500 interval 10
        event-control "oam" 2001 generate throttle
        event-control "ospf" 2001 suppress
        event-control "ospf" 2003 generate cleared
        event-control "ospf" 2014 generate critical
...
------------------------------------------------
```

7.9.4.6 Configuring a log filter

Use the commands in the following context to configure a log filter.

```
configure log filter
```

The following example shows a log filter configuration.

Example: MD-CLI

```
[ex:/configure log]
A:admin@node-2# info
    file "1" {
        description "This is our log file."
        rollover 600
        retention 24
        compact-flash-location {
            primary cf1
        }
    }
    filter "1" {
        description "This is a sample filter."
        default-action drop
        named-entry "1" {
            action forward
            match {
                application {
                    eq mirror
                }
                severity {
                    eq critical
                }
            }
        }
    }
    log-id "2" {
        admin-state disable
        description "This is a test log file."
        filter "1"
```

```
        source {
            main true
            security true
        }
        destination {
            file "1"
        }
    }
```

Example: classic CLI

```
A:node-2>config>log# info
#------------------------------------------
echo "Log Configuration "
#------------------------------------------
        file-id 1 name "1"
            description "This is our log file."
            location cf1:
            rollover 600 retention 24
        exit
        filter 1 name "1"
            default-action drop
            description "This is a sample filter."
            entry 1
                action forward
                match
                    application eq "mirror"
                    severity eq critical
                exit
            exit
        exit
...
        log-id 2 name "2"
            shutdown
            description "This is a test log file."
            filter 1
            from main security
            to file 1
        exit
...
------------------------------------------
```

7.9.4.7 Configuring an SNMP trap group

The associated log ID does not have to be configured before an SNMP trap group can be created; however, the SNMP trap group must exist before the log ID can be configured to use it.

Example: Basic SNMP trap group configuration (MD-CLI)

```
[ex:/configure log]
A:admin@node-2# info
    log-id "2" {
        description "This is a test log file."
        filter "1"
        source {
            main true
            security true
        }
        destination {
            snmp {
```

```
            }
        }
    }
```

Example: SNMP trap group, log, and interface configuration (MD-CLI)

```
[ex:/configure log]
A:admin@node-2# info
    snmp-trap-group "2" {
        trap-target "ops-mon-4" {
            address 10.10.10.104
            version snmpv2c
            notify-community "warnings-12a7"
        }
    }
```

Example: Basic SNMP trap group configuration (classic CLI)

```
A:node-2>config>log# info
----------------------------------------------
...
snmp-trap-group 2
    trap-target "ops-mon-4" address 10.10.10.104 snmpv2c notify-community "warnings-12a7"
exit
...
log-id 2
    description "This is a test log file."
    filter 1
    from main security
    to snmp
exit
...
----------------------------------------------
```

Example: SNMP trap group, log, and interface configuration (classic CLI)

```
A:node-2>config>log# snmp-trap-group 2
A:node-2>config>log>snmp-trap-group# info
----------------------------------------------
      trap-target "xyz-test" address xx.xx.x.x snmpv2c notify-community "xyztesting"
      trap-target "test2" address xx.xx.xx.x snmpv2c notify-community "xyztesting"
----------------------------------------------
*A:node-2>config>log>log-id# info
----------------------------------------------
      from main
      to snmp
----------------------------------------------
*A:node-2>config>router# interface xyz-test
*A:node-2>config>router>if# info
----------------------------------------------
      address xx.xx.xx.x/24
      port 1/1/1
----------------------------------------------
```

7.9.4.7.1 Setting the replay option

In the following example, the **replay** command option was set by an SNMP SET request for the trap-target address 10.10.10.3 which is bound to port-id 1/1/1.

Example: MD-CLI

```
[ex:/configure log snmp-trap-group "44"]
A:admin@node-2# info
    trap-target "test2" {
        address 10.20.20.5
        version snmpv2c
        notify-community "xyztesting"
    }
    trap-target "xyz-test" {
        address 10.10.10.3
        version snmpv2c
        notify-community "xyztesting"
        replay true
    }
```

Example: classic CLI

```
A:node-2>config>log>snmp-trap-group 44
A:node-2>config>log>snmp-trap-group# info
----------------------------------------------
trap-target "xyz-test" address 10.10.10.3 snmpv2c notify-community "xyztesting"
replay
trap-target "test2" address 10.20.20.5 snmpv2c notify-community "xyztesting"
----------------------------------------------
A:node-2>config>log>snmp-trap-group#
```

Use the following command to display the SNMP trap group log:

```
show log snmp-trap-group
```

Output example

In the following output, the **Replay** field changed from disabled to enabled.

```
===============================================================================
SNMP Trap Group 44
===============================================================================
Description : none
-------------------------------------------------------------------------------
Name        : xyz-test
Address     : 10.10.10.3
Port        : 162
Version     : v2c
Community   : xyztesting
Sec. Level  : none
Replay      : enabled
Replay from : n/a
Last replay : never
-------------------------------------------------------------------------------
Name        : test2
Address     : 10.20.20.5
Port        : 162
Version     : v2c
Community   : xyztesting
Sec. Level  : none
Replay      : disabled
Replay from : n/a
Last replay : never
===============================================================================
```

Because no events are waiting to be replayed, the log displays as before. Use the following command to display the log.

```
show log log-id
```

Output example

```
===============================================================================
Event Log 44
===============================================================================
SNMP Log contents  [size=100   next event=3819  (wrapped)]

3818 2008/04/22 23:35:39.89 UTC WARNING: SYSTEM #2009 Base IP
"Status of vRtrIfTable: router Base (index 1) interface xyz-test (index 35) changed
administrative state: inService, operational state: inService"

3817 2008/04/22 23:35:39.89 UTC WARNING: SNMP #2005 Base xyz-test
"Interface xyz-test is operational"

3816 2008/04/22 23:35:39.89 UTC WARNING: SNMP #2005 Base 1/1/1
"Interface 1/1/1 is operational"

3815 2008/04/22 23:35:39.71 UTC WARNING: SYSTEM #2009 Base CHASSIS
"Status of Mda 1/1 changed administrative state: inService, operational state:
 inService"

3814 2008/04/22 23:35:38.88 UTC MINOR: CHASSIS #2002 Base Mda 1/2
"Class MDA Module : inserted"

3813 2008/04/22 23:35:38.88 UTC MINOR: CHASSIS #2002 Base Mda 1/1
```

7.9.4.7.2 Disabling the SNMP notification outgoing port

Administratively disabling the port to which a trap-target address is bound removes the route to that trap target from the route table. When the SNMP module receives notification of this event, it marks the trap target as inaccessible and saves the sequence ID of the first SNMP notification that is missed by the trap target.

The following example shows how to disable the port and perform a log event test.

Example: Disable the outgoing port and perform a log event test (MD-CLI)

```
[ex:/configure]
A:admin@node-2# port 1/1/1 admin-state disable

[ex:/configure]
A:admin@node-2# commit

[ex:/configure]
A:admin@node-2# exit
INFO: CLI #2056: Exiting private configuration mode

[/]
A:admin@ node-2# tools perform log test-event
```

Example: Disable the outgoing port and perform a log event test (classic CLI)

```
A:node-2# configure port 1/1/1 shutdown
```

```
A:node-2# tools perform log test-event
```

Use the following command to display the SNMP trap group log.

```
show log snmp-trap-group
```

The following output example shows the Replay from field is updated with the sequence ID of the first event that is replayed when the trap-target address is added back to the route table.

Output example: SNMP trap group log

```
===============================================================================
SNMP Trap Group 44
===============================================================================
Description : none
-------------------------------------------------------------------------------
Name        : xyz-test
Address     : 10.10.10.3
Port        : 162
Version     : v2c
Community   : xyztesting
Sec. Level  : none
Replay      : enabled
Replay from : event #3819
Last replay : never
-------------------------------------------------------------------------------
Name        : test2
Address     : 10.20.20.5
Port        : 162
Version     : v2c
Community   : xyztesting
Sec. Level  : none
Replay      : disabled
Replay from : n/a
Last replay : never
===============================================================================
```

The following example shows event log output with trap targets that are not accessible and waiting for notification replay as well as the sequence ID of the first notification that is replayed.

Note: If there are more missed events than the log size, the replay actually starts from the first available missed event.

Output example: SNMP event log

```
===============================================================================
Event Log 44
===============================================================================
SNMP Log contents  [size=100   next event=3821  (wrapped)]
Cannot send to SNMP target address 10.10.10.3.
Waiting to replay starting from event #3819

3820 2008/04/22 23:41:28.00 UTC INDETERMINATE: LOGGER #2011 Base Event Test
"Test event has been generated with system object identifier tmnxModelSR12Reg.
System description: TiMOS-B-0.0.private both/i386 Nokia 7750 SR Copyright (c)
2000-2016 Nokia. All rights reserved. All use subject to applicable license
agreements. Built on Tue Apr 22 14:41:18 PDT 2008 by test123 in /test123/ws/panos/
main"

3819 2008/04/22 23:41:20.37 UTC WARNING: MC_REDUNDANCY #2022 Base operational state
```

```
 of peer chan*
"The MC-Ring operational state of peer 2.2.2.2 changed to outOfService."

3818 2008/04/22 23:35:39.89 UTC WARNING: SYSTEM #2009 Base IP
"Status of vRtrIfTable: router Base (index 1) interface xyz-test (index 35) changed
administrative state: inService, operational state: inService"

3823 2008/04/22 23:41:49.82 UTC WARNING: SNMP #2005 Base xyz-test
"Interface xyz-test is operational"
```

7.9.4.7.3 Re-enabling the in-band port

When you re-enable the in-band port to which a trap-target address is bound, the route to that trap target is re-added to the route table. When the SNMP trap module is notified of this event, it resends the notifications that were missed while there was no route to the trap-target address.

Use the following commands to enable a port and perform a log event test:

- **MD-CLI**

```
configure port admin-state enable
tools perform log test-event
```

- **classic CLI**

```
configure port no shutdown
tools perform log test-event
```

Use the following command to display the SNMP trap group log.

```
show log snmp-trap-group log-id-or-log-name
```

After the notifications are replayed, the Replay from field indicates n/a because there are no more notifications waiting to be replayed and the Last replay field timestamp has been updated.

Output example

```
===============================================================================
SNMP Trap Group 44
===============================================================================
Description : none
-------------------------------------------------------------------------------
Name        : xyz-test
Address     : 10.10.10.3
Port        : 162
Version     : v2c
Community   : xyztesting
Sec. Level  : none
Replay      : enabled
Replay from : n/a
Last replay : 04/22/2008 18:52:36
-------------------------------------------------------------------------------
Name        : test2
Address     : 10.20.20.5
Port        : 162
Version     : v2c
Community   : xyztesting
Sec. Level  : none
Replay      : disabled
```

```
Replay from : n/a
Last replay : never
===============================================================================
```

A display of the event log shows that it is no longer waiting to replay notifications to one or more of its trap target addresses. An event message has been written to the logger that indicates the replay to the trap-target address has happened and displays the notification sequence ID of the first and last replayed notifications.

Output example

```
===============================================================================
Event Log 44
===============================================================================
SNMP Log contents  [size=100   next event=3827  (wrapped)]

3826 2008/04/22 23:42:02.15 UTC MAJOR: LOGGER #2015 Base Log-id 44
"Missed events 3819 to 3825 from Log-id 44 have been resent to SNMP notification
 target address 10.10.10.3."

3825 2008/04/22 23:42:02.15 UTC INDETERMINATE: LOGGER #2011 Base Event Test
"Test event has been generated with system object identifier tmnxModelSR12Reg.
System description: TiMOS-B-0.0.private both/i386 Nokia 7750 SR Copyright (c)
2000-2016 Nokia.
All rights reserved. All use subject to applicable license agreements.
Built on Tue Apr 22 14:41:18 PDT 2008 by test123 in /test123/ws/panos/main"

3824 2008/04/22 23:41:49.82 UTC WARNING: SYSTEM #2009 Base IP
"Status of vRtrIfTable: router Base (index 1) interface xyz-test (index 35) changed
 administrative state: inService, operational state: inService"

3823 2008/04/22 23:41:49.82 UTC WARNING: SNMP #2005 Base xyz-test
"Interface xyz-test is operational"
```

7.9.4.8 Configuring a syslog target

A valid syslog ID must exist to send log events to a syslog target host.

The following example shows a syslog configuration.

Example: MD-CLI

```
[ex:/configure log]
A:admin@node-2# info
    syslog "1" {
        description "This is a syslog file."
        address 10.10.10.104
        facility user
        severity warning
    }
```

Example: classic CLI

```
A:node-2>config>log# info
----------------------------------------------
...
        syslog 1 name "1"
            description "This is a syslog file."
            address 10.10.10.104
```

```
            facility user
            level warning
        exit
...
-----------------------------------------------
```

7.9.4.9 Modifying a log file

A log file configuration can be modified.

Example: MD-CLI

The following example shows a current log file configuration.

```
[ex:/configure log]
A:admin@node-2# info
    log-id "2" {
        description "This is a test log file."
        filter "1"
        source {
            main true
            security true
        }
        destination {
            file "1"
        }
    }
```

The following example shows modifications to the log file configuration.

```
[ex:/configure]
A:admin@node-2# log log-id 2

[ex:/configure log log-id "2"]
A:admin@node-2# description "Chassis log file"

[ex:/configure log log-id "2"]
A:admin@node-2# filter 2

[ex:/configure log log-id "2"]
A:admin@node-2# destination file 2

[ex:/configure log log-id "2"]
A:admin@node-2#
```

The following example shows the results of the modifications to the log file configuration.

```
*[ex:/configure log]
A:admin@node-2# info
    log-id "2" {
        description "Chassis log file."
        filter "2"
        source {
            security true
        }
        destination {
            file "1"
        }
    }
```

Example: classic CLI

The following example shows a current log file configuration.

```
A:node-2>config>log>log-id# info
----------------------------------------------
...
log-id 2 name "2"
            description "This is a test log file."
            filter 1
            from main security
            to file 1
exit
...
----------------------------------------------
```

The following example shows modifications to the log file configuration.

```
*A:node-2>config# log
*A:node-2>config>log# log-id 2
*A:node-2>config>log>log-id# description "Chassis log file."
*A:node-2>config>log>log-id# filter 2
*A:node-2>config>log>log-id# from security
*A:node-2>config>log>log-id# exit
```

The following example shows the results of the modifications to the log file configuration.

```
A:node-2>config>log# info
----------------------------------------------
...
log-id 2 name "2"
            description "Chassis log file."
            filter 2
            from security
            to file 1
exit
...
----------------------------------------------
```

7.9.4.10 Deleting a log file

Use the following command to delete a log file:

- **MD-CLI**
  It is not necessary to disable the log ID before you delete it. Also, you can use the **delete** command in any context.

  ```
  delete
  ```

- **classic CLI**
  You must shutdown the log ID before you delete it.

  ```
  configure log log-id shutdown
  configure log no log-id 2
  ```

The following example shows how to delete a log file.

Example: MD-CLI

```
[ex:/configure log log-id "2"]
A:admin@node-2# info
    description "filter "1001"
    destination {
        file "50"
    }

*[ex:/configure log]
A:admin@node-2# delete log-id 2
```

Example: classic CLI

```
A:node-2>config>log# info
----------------------------------------------
file-id name "1"
            description "LocationTest."
            location cf1:
            rollover 600 retention 24
        exit
...
log-id name "2"
            description "Chassis log file."
            filter 2
            from security
            to file 1
exit
...
*A:node-2>config>log# log-id 2
*A:node-2>config>log>log-id# shutdown
*A:node-2>config>log>log-id# exit
*A:node-2>config>log# no log-id 2
```

7.9.4.11 Modifying a log file ID

Example: MD-CLI

The following example shows the current log file configuration.

```
[ex:/configure log]
A:admin@node-2# info
    file "1" {
        description "This is a log file."
        rollover 600
        retention 24
        compact-flash-location {
            primary cf1
        }
    }
```

The following example shows the modifications to the current log file configuration.

```
*[ex:/configure log]
A:admin@node-2# file 1

*[ex:/configure log file "1"]
A:admin@node-2# description "LocationTest."
```

```
*[ex:/configure log file "1"]
A:admin@node-2# rollover 2880

*[ex:/configure log file "1"]
A:admin@node-2# retention 500

*[ex:/configure log file "1"]
A:admin@node-2# compact-flash-location

*[ex:/configure log file "1" compact-flash-location]
A:admin@node-2# primary cf2
```

The following example shows the results of the modifications to the log file configuration.

```
[ex:/configure log]
A:admin@node-2# info
    file "1" {
        description "LocationTest."
        rollover 2880
        retention 500
        compact-flash-location {
            primary cf2
        }
    }
```

Example: classic CLI

The following example shows the current log file configuration.

```
A:node-2>config>log# info
----------------------------------------
        file-id name "1"
            description "This is a log file."
            location cf1:
            rollover 600 retention 24
        exit
----------------------------------------------
```

The following example shows modifications to the log file configuration.

```
*A:node-2>config>log# file-id 1
*A:node-2>config>log>file-id# description "LocationTest."
*A:node-2>config>log>file-id# location cf2:
*A:node-2>config>log>file-id# rollover 2880 retention 500
*A:node-2>config>log>file-id# exit
```

The following example shows the results of the modifications to the log file configuration.

```
A:node-2>config>log# info
----------------------------------------------
...
file-id name "1"
    description "LocationTest."
    location cf2:
    rollover 2880 retention 500
exit
...
----------------------------------------------
```

7.9.4.12 Modifying a syslog ID

You can modify the syslog ID for a log. All references to the syslog ID must be deleted before the syslog ID can be removed.

Example: MD CLI

The following example shows modifications to the syslog 1 configuration.

```
[pr:/configure log]
A:admin@Dut-G# syslog 1

*[pr:/configure log syslog "1"]
A:admin@Dut-G# description "Test syslog"

*[pr:/configure log syslog "1"]
A:admin@Dut-G# address 10.10.0.91

*[pr:/configure log syslog "1"]
A:admin@Dut-G# facility mail

*[pr:/configure log syslog "1"]
A:admin@Dut-G# severity info

*[pr:/configure log syslog "1"]
```

The following example shows the results of the modifications to the syslog 1 configuration.

```
[ex:/configure log syslog]
A:admin@node-2# info
...
    syslog "1" {
        description "Test syslog"
        address 10.10.0.91
        facility mail
        severity info
        }
    }
```

Example: classic CLI

The following example shows modifications to the syslog 1 configuration.

```
*A:node-2>config# log
*A:node-2>config>log# syslog 1
*A:node-2>config>log>syslog$ description "Test syslog."
*A:node-2>config>log>syslog# address 10.10.0.91
*A:node-2>config>log>syslog# facility mail
*A:node-2>config>log>syslog# level info
```

The following example shows the modified syslog configuration output.

```
A:node-2>config>log# info
----------------------------------------------
...
        syslog 1 name "1"
            description "Test syslog."
            address 10.10.10.91
            facility mail
```

```
            level info
        exit
...
-----------------------------------------------
```

7.9.4.13 Deleting an SNMP trap group

Use the following commands to delete SNMP trap groups:

- **MD-CLI**

```
configure log snmp-trap-group delete trap-target
configure log delete snmp-trap-group
```

- **classic CLI**

```
configure log snmp-trap-group no trap-target
configure log no snmp-trap-group
```

Example: classic CLI

The following example shows an SNMP trap group configuration.

```
A:node-2>config>log# info
----------------------------------------------
...
       snmp-trap-group name "10"
           trap-target "ops-mon-4" address 10.10.10.104 snmpv2c notify-community
 "warnings-12a7"
       exit
...
----------------------------------------------
```

The following example shows deleting the trap target and SNMP trap group.

```
*A:node-2>config>log# snmp-trap-group
*A:node-2>config>log>snmp-trap-group# no trap-target ops-mon-4
*A:node-2>config>log>snmp-trap-group# exit
*A:node-2>config>log# no snmp-trap-group 10
```

7.9.4.14 Modifying a log filter

You can modify the configuration of a log filter.

Example: MD- CLI

The following example shows a log filter configuration.

```
[ex:/configure log]
A:admin@node-2# info
...
    filter "1" {
        description "This is a sample filter with default action drop."
        default-action drop
        }
    }
```

```
...
```

The following example shows the modifications applied to the log filter configuration.

```
*[ex:/configure]
A:admin@node-2# log filter 1

*[ex:/configure log filter "1"]
A:admin@node-2# description "This filter allows forwarding"

*[ex:/configure log filter "1"]
A:admin@node-2# default-action forward
```

The following example shows the results of the modifications to the log filter configuration.

```
A:node-2>config>log>filter# info
----------------------------------------
...
        filter name "1"
            description "This filter allows forwarding"
             default-action forward
            exit
        exit
...
----------------------------------------
```

Example: classic CLI

The following example shows a log filter configuration.

```
A:node-2>config>log# info
#------------------------------------------
echo "Log Configuration "
#------------------------------------------
...
        filter name "1"
            default-action drop
            description "This is a sample filter."
            entry 1
                action forward
                match
                    application eq "mirror"
                    severity eq critical
                exit
            exit
        exit
...
------------------------------------------
```

The following example shows the modifications applied to the log filter configuration.

```
A:node-2>config# log
*A:node-2>config>log# filter 1
*A:node-2>config>log>filter# description "This allows <n>."
*A:node-2>config>log>filter# default-action forward
*A:node-2>config>log>filter# entry 1
*A:node-2>config>log>filter>entry$ action drop
*A:node-2>config>log>filter>entry# match
*A:node-2>config>log>filter>entry>match# application eq user
*A:node-2>config>log>filter>entry>match# number eq 2001
*A:node-2>config>log>filter>entry>match# no severity
```

```
*A:node-2>config>log>filter>entry>match# exit
```

The following example shows the results of the modifications to the log filter configuration.

```
A:node-2>config>log>filter# info
----------------------------------------
...
        filter name "1"
            description "This allows <n>."
            entry 1
                action drop
                match
                    application eq "user"
                    number eq 2001
                exit
            exit
        exit
...
----------------------------------------
```

7.9.4.15 Modifying event control configuration

Example: MD-CLI

The following example shows a current event control configuration.

```
[ex:/configure log]
A:admin@node-2# info
    log-events {
        bgp event tmnxOspfVirtIfStateChange {
            generate false
            throttle false
        }
        ospf event tmnxOspfVirtNbrStateChange {
            severity cleared
            throttle false
        }
        ospf event tmnxOspfLsdbOverflow {
            severity critical
            throttle false
        }
    }
    throttle-rate {
        limit 500
        interval 10
    }
```

The following example shows a modification to the event control configuration.

```
*[ex:/configure log]
A:admin@node-2# log-events bgp 2014 suppress
```

The following example shows the modified event control configuration.

```
*[ex:/configure log]
A:admin@node-2# event-control ospf 2014 suppress
```

Example: classic CLI

The following example shows a current event control configuration.

```
A:node-2>config>log# info
----------------------------------------------
...
event-control "bgp" 2014 generate critical
...
----------------------------------------------
```

The following example shows a modification to the event control configuration.

```
*A:node-2>config# log
*A:node-2>config>log# event-control bgp 2014 suppress
```

The following example shows the modified event control configuration.

```
*A:node-2>:config# log
*A:node-2>config>log# event-control ospf 2014 suppress
```

7.9.4.16 Returning to the default event control configuration

Use the following command to delete modified log event options and return them to the default values:

- **MD-CLI**

```
configure log log-events delete event option
```

- **classic CLI**

```
configure log no event-control application [event-name | event-number]
```

The following example shows the command usage.

Example: MD-CLI

```
[ex:/configure log log-events]
A:admin@node-2# delete snmp event authenticationFailure
```

Example: classic CLI

```
A:node-2>config>log# no event-control "bgp" 2001
```

8 Public key infrastructure

8.1 X.509v3 certificate overview

X.509v3 is an ITU-T standard which consists of a hierarchical system of Certificate Authorities (CAs) that issue certificates that bind a public key to particular entity's identification. The entity's identification could be a distinguished name or an alternative name such as FQDN or IP address.

An end entity is an entity that is not CA. For example an end entity can be a web server, a VPN client, or a VPN gateway.

A CA issues a certificate by signing an entity's public key with its own private key. A CA can issue certificates for an end entity as well as for another CA. In the case when a CA certificate is issued by itself (signed by its own private key), then this CA is called the root CA. Thus, an end entity's certificate could be issued by the root CA or by a subordinate CA (this is issued by another subordinate CA or root CA). When there are multiple CA involved, it is called a chain of CAs.

A PKI also includes the mechanism for revoking certificates because of reasons such as a compromised private key.

The certificate can be used for different purposes. One purpose is authentication. Typically certificate authentication functions as following:

- The system trusts a CA as trust anchor CA (which typically is a root CA). This means that all certificates issued by a trust anchor CA, or the certificates issued by a sub CA issued by the trust anchor CA, are consider trusted.
- A peer to be authenticated presents its certificate along with a signature over some shared data between the peer and system, which is signed by using a private key.
- The signature is verified by using the public key in the certificate, and the certificate is verified; that is, it is issued by the trust anchor CA or a sub-CA in a chain up to the trust anchor CA. The system can also check if the peer's certificate has been revoked. Only when all these verifications succeed, then the certificate authentication succeeds.

8.1.1 SR OS X.509v3 certificate support

The SR OS PKI implementation supports the following features:

- Supported public key algorithm: RSA/DSA/ECDSA
- Certificate enrollment includes:
  - Locally generated RSA/DSA/ECDSA key
  - Offline enrollment via PKCS#10
  - Online enrollment via Certificate Management Protocol version 2 (CMPv2)
  - Online enrollment via Enrollment over Secure Transport protocol (EST)
- Support CA chain
- Certificate revocation check:

  - CRL for both EE (End Entity) and CA certificate
  - OCSP for EE certificate only

8.1.2 Local storage

SR OS requires the following objects to be stored locally as file:

- CA Certificate
- CRL
- certificate of the system
- key of the system

Before they can be used by SR OS, use the following command to import the preceding objects:

- **MD-CLI**

```
admin system security pki import
```

- **classic CLI**

```
admin certificate import
```

The import process converts the format of input file to DER, encrypts it, and saves it in the `cf3:/system-pki` directory.

The imported file can also be exported as one to use in the specified format by means of the following command:

- **MD-CLI**

```
admin system security pki export
```

- **classic CLI**

```
admin certificate export
```

The preceding commands support the following formats:

- Certificates can be imported or exported using the following formats:
  - PKCS#12
  - PKCS#7 (DER and PEM)
  - PEM
  - DER

  If there are multiple certificates in the file, only the first one is used.
- Key pairs can be imported or exported by using the following formats:
  - PKCS#12 (must along with certificate)
  - PEM
  - DER

- CRL can be imported or exported by using the following formats:
  - PKCS#7 (DER and PEM)
  - PEM
  - DER
- PKCS#12 file can be encrypted with a password.

8.1.3 CA-profile

SR OS stores the CA-related configuration in a Certificate Authority profile (CA-profile), which contains the following configurations:

- name and description
- CA's certificate (an imported certificate)
- CA's CRL (an imported CRL)
- revocation check method (specifies the way CA checks the revocation status of the certificate it issued)
- CMPv2 (a CMPv2 server related configurations)
- OCSP (an OCSP responder related configurations)

When a CA profile is enabled, the system loads the specified CA certificate and CRL into memory and performs the following checks:

- **for CA certificate**
  - All non-optional fields defined in section 4.1 of RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, exist and conform to the format defined in RFC 5280.
  - Check that the value of the version field is 0x2.
  - Check that the certificate of the Validity field is still in the validity period.
  - The X509 Basic Constraints extension exists and the CA Boolean is True.
  - If the Key Usage extension exists, at least the keyCertSign is asserted.
- **for CRL**
  - All non-optional fields defined in section 5.1 of RFC 5280 must exist and conform to the format defined in RFC 5280.
  - If the version field exists, the value must be 0x1.
  - The delta CRL Indicator must not be present (Delta CRL is not supported).
  - The CRL must be signed by the configured CA certificate.

A CRL, by default, is required to enable the CA profile. However, you can change the revocation check method configuration to make this optional. For information about the revocation check method configuration, see Certificate revocation check.

8.1.4 CA chain computation

In case of verifying a certificate with a CA or a chain of CAs, the system needs to identify the issuer CA of the certificate in question. The SR OS looks through all configured ca-profiles to find the issuer CA. The following is the method system used to find the issuer CA:

- The issuer CA's certificate subject must match the issuer field of the certificate in question.
- If present, the authority key identifier of the certificate in question must match the subject key identifier of the issuer CA's certificate.
- If present, the key usage extension of the issuer CA's certificate must permit certificate signing.

8.1.5 Certificate enrollment

About this task

SR OS supports the following enrollment methods:

- offline method via PKCS#10
- online method via CMPv2
- online method via EST

The offline method works as follows.

For CMPv2-based enrollment, see CMPv2. For EST-based enrollment, see Enrollment over Secure Transport.

Procedure

Step 1. Generate a key pair using the following command:

- **MD-CLI**

```
admin system security pki generate-keypair
```

- **classic CLI**

```
admin certificate gen-keypair
```

Example

MD-CLI

```
[/admin system security pki]
A:admin@node-2# generate-keypair cf3:/segw.key rsa-key-size 2048
```

Example

classic CLI

```
A:node-2# admin certificate gen-keypair cf3:/segw.key size 2048 type rsa
```

Step 2. Generate a PKCS#10 certificate signing request with the key generated in the preceding step using the following command:

- **MD-CLI**

```
admin system security pki generate-csr
```

- **classic CLI**

```
admin certificate gen-local-cert-req
```

The user specifies the subject of certificate request and optionally can also specify a FQDN or an IP address as SubjectAltName.

Example

MD-CLI

```
/admin system security pki]
A:admin@Dnode-2# generate-csr key-url cf3:/segw.key output-url cf3:/segw.pkcs10
 subject-dn C=US,ST=CA,O=ALU,CN=SeGW domain-name segw-1.alu.com
```

Example

classic CLI

```
A:node-2# admin certificate gen-local-cert-req keypair cf3:/segw.key subject-dn C=
US,ST=CA,O=ALU,CN=SeGW domain-name segw-1.alu.com file cf3:/segw.pkcs10
```

Step 3. Import the key file using the following command:

- **MD-CLI**

```
admin system security pki import
```

- **classic CLI**

```
admin certificate import
```

Example

MD-CLI

```
[/admin system security pki]
A:admin@node-2# import type key format der input-url cf3:/segw.key output-file sew.key
```

Example

classic CLI

```
A:node-2# admin certificate import type key input cf3:/segw.key output segw.key format
 der
```

Step 4. Because the key is imported, remove the key file generated in the first step for security reasons.

Step 5. Send the PKCS#10 file to CA via an offline method such as e-mail.

Step 6. CA signs the request, and returns the certificate.

Step 7. Import the resulting certificate using the following command:

- **MD-CLI**

```
admin system security pki import
```

- **classic CLI**

```
admin certificate import
```

Example

MD-CLI

```
[/admin system security pki]
A:admin@node-2# import type certificate format pem input-url cf3:/segw.cert output-
file segw.cert
```

Example

classic CLI

```
A:node-2# admin certificate import type cert input cf3:/segw.cert output segw.cert
 format pem
```

8.1.6 Certificate revocation check

A revocation check is a process to see if a certificate has been revoked by the issuer CA.

The SR OS supports two methods for certificate revocation check:

- CRL – can be used for both EE and CA certificate checks
- OCSP – can only be used for an EE certificate

The use of a revocation check for an EE certificate is application specific. With an IPsec application, users can configure multiple check methods with a priority order for an EE certificate. A primary method, a secondary method, and a default result can be configured:

- in the MD-CLI, with the commands under **ipsec-tunnel key-exchange dynamic cert status-verify** or **ipsec-gateway cert status-verify**
- in the classic CLI, with the commands under **ipsec-tunnel dynamic-keying cert status-verify** or **ipsec-gw cert status-verify**

The primary and secondary method can be either OCSP or CRL. The default result is either **good** or **revoked**. If the system cannot get an answer from the primary method, it falls back to the secondary method. If the secondary method also does not return an answer, the system uses the default result.

By default, the system uses CRL to check the revocation status of a certificate, whether it is an end entity certificate or a CA certificate. This makes CRL a mandatory configuration in the CA profile.

The **revocation-check** command in the **ca-profile** can change this behavior, with **revocation-check crl-optional** configured using the following command.

```
configure system security pki ca-profile revocation-check {crl-optional}
```

When a user enables the CA profile, the system tries to load the configured CRL (specified by the **crl-file** command). But, if the system fails to load it for the following reasons, the system still keeps the CA profile operationally up, but treats the CRL as nonexistent:

- The CRL file does not exist.
- The CRL is not properly encoded, possibly because of an interrupted file transfer.
- The CRL is not signed by the CA certificate configured in the CA profile.
- The CRL version is wrong.
- The CRL expired or is not yet valid.

If the IPsec application needs to use the CRL of a specific **ca-profile** to check the revocation status of an end entity certificate and the CRL is nonexistent because of the preceding reasons, the system treats it as unable to get an answer from the CRL and falls back to the secondary **status-verify** method or the **default-result** configured under the following contexts:

- in the MD-CLI, with the commands under **ipsec-tunnel key-exchange dynamic cert status-verify** or **ipsec-gateway cert status-verify**
- in the classic CLI, with the commands under **ipsec-tunnel dynamic-keying cert status-verify** or **ipsec-gw cert status-verify**

If the system needs to check the revocation of a CA certificate in the certificate chain, and if the CRL is nonexistent because of the preceding reasons, the system skips checking the revocation status of the CA certificate. For example, CA1 is issued by CA2, if the **revocation-check** command of CA2 is configured to **crl-optional** and the CRL for CA2 is nonexistent, the system does not check the certificate revocation status of CA1 and considers it **good**.

In the classic CLI, the user must disable the **ca-profile** to change the **revocation-check** configuration.

For more information about OCSP, see OCSP.

8.1.7 Certificate/CRL expiration warning

The system can optionally generate a warning message before a certificate or a CRL expires. Use the following commands to configure the amount of time before expiration for the certificate or CRL respectively.

```
configure system security pki certificate-expiration-warning
configure system security pki crl-expiration-warning
```

The warning messages can also be optionally repeated at a configured interval. For more information about the warning messages, see the corresponding command descriptions in the following guides:

- *7705 SAR Gen 2 MD-CLI Command Reference Guide*
- *7705 SAR Gen 2 Classic CLI Command Reference Guide*

If a configured EE certificate expires, the system does not bring down an established IPsec tunnel or bring down an IPsec gateway; however, future certificate authentication fails.

If a CA certificate expires, the system brings the CA profile operationally down. This does not affect established tunnels; however, future certificate authentication that uses the CA profile fails.

8.1.8 Certificate, CRL, or key cache

Configured certificates, CRLs, and keys are cached in memory before they are used by the system. The following information describes the use of the certificates, CRLs, and keys:

- Every certificate, CRL, or key has one cache copy system wide.
- For a CA certificate and CRL, the cache is created when there is a CA profile and when it is administratively enabled and removed.
- For an IPsec tunnel or IPsec gateway using legacy **cert** and **key** configurations, the cache is created only when the first tunnel using it is in an administratively enabled state, and it is cleared when the last tunnel that used it is administratively disabled.
- For an IPsec tunnel or IPsec gateway using **cert-profile**, the cache is created when the first **cert-profile** using it is in an administratively disabled state, and the cache is removed when the last **cert-profile** that used it is in an administratively disabled state.
- If a certificate or key is configured with both a **cert-profile** and legacy **cert** or **key** command, the cache is created when the first object (a **ipsec-gw**, **ipsec-tunnel** or **cert-profile**) using it is administratively enabled and the cache is removed when the last object using it is administratively disabled.

To update a certificate or key without administratively disabling the CA profile, IPsec tunnel, or IPsec gateway, use the following command to reload the certificate and key cache:

- **MD-CLI**

```
admin system security pki reload
```

  See the *7705 SAR Gen 2 MD-CLI Command Reference Guide* for more information.

- **classic CLI**

```
admin certificate reload
```

  See the *7705 SAR Gen 2 Classic CLI Command Reference Guide* for more information.

8.1.9 Auto CRL update

SR OS provides an automatic mechanism to update a CRL file. The system tries to download the CRL from a list of configured HTTP URLs and replaces the existing CRL file when a qualified CRL is successfully downloaded. A qualified CRL is a valid CRL signed by the CA and is more recent than the existing CRL. To determine if a downloaded CRL is more recent than an existing CRL, the system compares the This-Update field of the CRL first. If they are the same, the system compares the CRL number extension if present.

The configured HTTP URL must point to a DER-encoded CRL file.

This features supports the following types of downloading schedules:

- **periodic**

  The system downloads a CRL periodically at the interval configured via the **periodic-update-interval** command. For example, if the **periodic-update-interval** is 1 day, the system downloads CRL every 1 day. The minimal periodic-update-interval is 1 hour.

- **next-update-based**

  The system downloads a CRL at the time = Next_Update_time_of_current_CRL minus pre-update-time. For example, if the Next-Update of current CRL is 2015-06-30 06:00 and pre-update-time is 1 hour, the system starts the download at 2015-06-30, 05:00.

The system allows up to eight URLs to be configured for a CA profile. When downloading begins, URLs are tried in order, and the first successfully downloaded qualified CRL is used to update the existing CRL. If

the downloading fails or the downloaded CRL is not qualified, the system moves to the next URL in the list. If all URLs in the list fail to return a qualified URL, the following occurs:

- In case of next-update-based schedule, the system waits for a configured **retry-interval** before retrying from the first URL in the list again.
- In case of periodic schedule, the system waits until the next scheduled time.

After administratively enabling a CA profile and with **auto-crl-update** is enabled, in a case where the CRL file does not exist or is expired or invalid, the system starts downloading immediately.

The system also provides an **admin** command for users to trigger downloading:

- **MD-CLI**

```
admin system security pki crl-update ca-profile ca-profile-name
```

- **classic CLI**

```
admin certificate crl-update ca ca-profile-name
```

However, it requires the user to administratively disable the **auto-crl-update** command using the following command:

- **MD-CLI**

```
configure system security pki ca-profile auto-crl-update admin-state disable
```

- **classic CLI**

```
configure system security pki ca-profile no auto-crl-update
```

HTTP transport can be over either IPv4 or IPv6.

This feature supports a base, management, or VPRN routing instance. VPLS management is not supported. For VPRN, the HTTP server port can only be 80 or 8080.

8.1.10 CMPv2

CMPv2, RFC 4210, *Internet X.509 Public Key Infrastructure Certificate Management Protocol* (*CMP*) is a protocol between a Certificate Authority (CA) and an end entity. It provides multiple certificate management functions like certificate enrollment, certificate update, and so on.

SR OS supports following CMPv2 operations:

- **initial registration**

  This is the process SR OS uses to enroll a certificate with a specific CA for the first time.

  – Public/private key pair must be pre-provisioned before enrollment by means of local generation or other methods.

  – Users can optionally include a certificate or certificate chain in the extraCerts field of the initial registration request.

- **key pair update**

  This is a process for SR OS to update an existing certificate because of reasons like refreshes key/cert before it expires or any other reason.

- **certificate update**

  This is a process where an initialized SR OS system obtains additional certificates.

- **polling**

  In some cases, the CA may not return the certificate immediately for reasons such as request processing needing manual intervention. In such cases, SR OS supports polling requests and responds as described in Section 5.3.22, Polling Request and Response, in RFC 4210, *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)*.

The following lists some implementation details:

- HTTP is the only supported transport protocol for CMPv2. HTTP 1.1 and 1.0 are supported and configurable.
- All CMPv2 messages sent by SR OS consist of only one PKI Message. The size of the sequence for PKI Messages are 1 in all cases.
- Both the password-based MAC and the public key-based signature CMPv2 message protection are supported.
- SR OS only allows one outstanding ir/cr/kur request for each CMPv2 server. The means that no new requests are allowed if a pending request is present.
- If a CMPv2 PKIConf message does not contain the extraCerts field and the system is configured to use extraCerts to verify the message, the extraCerts field from a previously received response message, such as the initial registration response or the key update response, is used.

8.1.11 Encryption of imported files

The following storage formats for imported certificates, keys, and CRLs:

- legacy, where only the imported key is encrypted
- enhanced secure, where:
  - The encryption algorithm is stronger than the legacy format.
  - Imported certificates and keys are both encrypted.
  - The internal key for encryption is chassis specific. In the case of VSR, the key is VM UUID specific.
  - A compressed format is used for imported CRL files to save space.

The legacy format is used in SR OS releases before Release 16.0.R6. The enhanced secure format is used for all imported files from Release 16.0.R6 and later. By default, the system loads an imported file in both legacy and enhanced secure formats.

Use the following command to configure the system to only load imported files in the enhanced secure format.

```
configure system security pki imported-format secure
```

Use the following command to convert imported files between the legacy format and the enhanced secure format:

- **MD-CLI**

```
admin system security pki convert-file
```

- **classic CLI**

```
admin certificate convert-file
```

8.1.12 Enrollment over Secure Transport

The Enrollment over Secure Transport (EST) protocol as described in RFC 7030, *Enrollment over Secure Transport*, is used to enroll a certificate from a Certificate Authority (CA). SR OS supports the following EST client-side operations:

- downloading a CA certificate (/cacert)
- enrolling a new certificate (/simpleenroll)
- renewing an existing certificate (/simplereenroll)

Use the commands in the following context to perform the EST client-side operations. Each operation requires an EST profile that contains the EST configuration:

- **MD-CLI**

```
admin system security pki est
```

- **classic CLI**

```
admin certificate est
```

The following option is supported for the SR OS client to authenticate the EST server. Use the following command to configure Explicit TA, which is referenced in the EST profile.

```
configure system security tls client-tls-profile trust-anchor-profile
```

No authentication is performed if this option is not configured.

The following options are supported for the EST server authentication to the SR OS client:

- Use the commands in the following contexts to achieve the client certificate authentication by configuring the certificate profile name for the client TLS profile referenced in the EST profile.

```
configure system security tls cert-profile
configure system security tls client-tls-profile
```

- Use the following command to configure HTTP authentication.
  - **MD-CLI**

```
configure system security pki est-profile http-authentication
```

  - **classic CLI**

```
configure system security pki est-profile http-auth
```

- Use the following command to configure the trust anchor profile name referenced in the EST profile.

```
configure system security tls client-tls-profile trust-anchor-profile
```

- No authentication is performed if the preceding options are not configured.

Configuring a URL for each EST message type

SR OS allows users to configure a URL for each EST message type in an EST profile. The following default URLs are used if the URL of the corresponding operation is not configured:

- cacerts: /.well-known/est/cacerts
- simpleenroll: /.well-known/est/simpleenroll
- simplereenroll: /.well-known/est/simplereenroll

Use the following command to configure the server path for the distribution of CA certificates:

- **MD-CLI**

```
configure system security pki est-profile server path cacerts
```

- **classic CLI**

```
configure system security pki est-profile request-path cacerts
```

Use the following command to configure the server path for the enrollment of clients:

- **MD-CLI**

```
configure system security pki est-profile server path simpleenroll
```

- **classic CLI**

```
configure system security pki est-profile request-path simpleenroll
```

Use the following command to configure the server path for the reenrollment of clients:

- **MD-CLI**

```
configure system security pki est-profile server path simplereenroll
```

- **classic CLI**

```
configure system security pki est-profile request-path simplereenroll
```

8.1.13 OCSP

Online Certificate Status Protocol (OCSP) (RFC 2560, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*) is used by SR OS applications to determine the (revocation) state of an identified certificate. Unlike CRL, which relies on checking against an off-line file, OCSP provides timely, on-line information about the revocation status of a certificate.

IPsec is the only supported application to use OCSP. With introduction of OCSP, the system supports both CRL and OCSP as the certificate revocation status checking method. For an ipsec-tunnel or ipsec-gw, the user could configure a primary method, a secondary method and a default result to achieve a hierarchical fallback mechanism. If the primary method fails to return a result, the system falls back to the secondary method. If the secondary method fails, the fall back proceeds to a default result.

The following lists implementation details:

- Only an OCSP client function is supported.
- HTTP is the only supported transport protocol.
- OCSP server access via management routing instance is not supported.
- SR OS does not sign an OCSP Request.
- The OCSP response must be signed. The system verifies the response by using the signer's certificate included in the response. If there is no such certificate, the CA certificate in the ca-profile is used.
- If a nextUpdate exists in the OCSP response, the system checks the current time <= nextUpdate. If yes, then the response is valid, otherwise the response is considered unreliable. The system moves to next revocation checking method.
- The revocation status result from a valid OCSP response is cached in the system.
- OCSP can only be used to verify the revocation status of the end-entity certificate. CRL is still needed for CA certificate's status verification.

8.1.14 Auto update certificate

SR OS supports automatic updating of an imported end-entity certificate by using an online enrollment protocol with CA. The following enrollment protocols are supported:

- CMPv2 (RFC 4210)
- EST (RFC 7030)

For each certificate that needs an automatic update, a **certificate-auto-update** command entry must be configured as well as the corresponding **certificate-update-profile** command. The **certificate-update-profile** command specifies the update behavior such as the enrollment protocol to use, the schedule type, and so on.

The following events may trigger an update:

- When the current time passes a user-specified deadline, the deadline can be configured as one of the schedule types in the **certificate-update-profile**:
  - **before-expiry** configures the time before the certificate expiration time
  - **after-issue** configures the time after certificate issue time
- When a **certificate-auto-update** entry is configured, and it is already time to do an update.

  If the certificate already expired:
  - for CMPv2, the update fails because CMPv2 does not allow using an expired certificate
  - for EST, if a different certificate is used for TLS authentication, the update is completed
- Manually, by using the following command:
  - **MD-CLI**

    ```
    admin system security pki update-certificate
    ```
  - **classic CLI**

    ```
    admin certificate update-cert
    ```

Note: This feature uses the UTC, not the local time.

The following shows the workflow of a certificate update:

1. A new key is generated.
   - If the following command is configured in the **certificate-update-profile**, the system generates a new key with the same type and the same length as the existing key.
     - **MD-CLI**

       ```
       configure system security pki certificate-update-profile same-as-existing-key
       ```

     - **classic CLI**

       ```
       configure system security pki certificate-update-profile key-generation same-as-existing-key
       ```

   - Otherwise, a new key is generated according to the key generation configuration.
2. Use the corresponding operation of the enrollment protocol specified in **certificate-update-profile** configuration to obtain a new certificate from the CA.
   - CMPv2 configures the key-update operation.
   - EST configures the renew (or /simplereenroll) operation.
3. After the configuration obtains a new certificate from the CA (step 3), import and replace the existing key and certificate file with the same filename. The existing key and certificate file are renamed by adding a ".previous" suffix. If there are existing "xxx.previous" files, they are removed. If either of the previous fails, the existing key and certificate are not impacted.
4. The application (for example, IPsec) that uses the certificate, reloads the key and certificate so that new key and certificate are used.
5. If step 1, step 2, or step 3 fails, the system waits for the retry interval specified in the **certificate-update-profile** to retry from step 1. If step 4 fails, then skip steps 1, 2, and 3 and then wait for the *retry-interval* to retry from step 4.

9 sFlow

9.1 sFlow overview

Some Layer 2 network deployments collect statistics on physical Ethernet ports and on Layer 2 interfaces at a high-frequency using a push model to, among others, monitor traffic, diagnose network issues, or provide billing. SR OS supports cflowd and XML accounting; however, those mechanisms are either Layer 3-specific, or focus on providing statistics at extremely large scale (therefore use a pull model and cannot support high-frequency counter updates). To meet the statistics collection requirements of such Layer 2 deployments, SR OS supports sFlow statistics export using sFlow version 5.

Note: sFlow implementation is targeted for low-scale deployments.

9.2 sFlow features

This section describes sFlow functionality supported in SR OS.

9.2.1 sFlow counter polling architecture

When sFlow is enabled on an SR OS router, the system takes upon a role of an sFlow network device as described in sFlow protocol version 5. A single sFlow agent can be configured for counter polling (flow sampling is not supported). There is no support for sub-agents.

The sFlow agent sends sFlow data to an operator-configured sFlow receiver. A single receiver is supported with configurable primary and backup IPv4 or IPv6 UDP destination sockets for redundancy (each sFlow packet exported is duplicated to both sockets when both are configured). The receiver's UDP sockets can be reachable either in-band or out-of-band (default) and must both be IPv4 or IPv6. An operator can also set the maximum size of the sFlow datagrams. Operators are expected to set this value to avoid IP fragmentation (Datagrams exceeding the specified size are fragmented before handed to IP layer).

The sFlow agent manages all sFlow data sources in the system. SR OS supports sFlow data that are physical ports. When a port is configured as an sFlow data source, counters for that port and all VPLS and Epipe SAPs on that port are collected and exported using sFlow (see sFlow record formats). Flow data sources can only be configured when an sFlow receiver is configured. To remove the sFlow receiver, all sFlow data sources must first be deconfigured at the port level.

Each data source is processed at a 15-second, non-configurable interval. If multiple data sources exist on a line card, the line card distributes the processing of each data source within a 15 second interval to avoid sFlow storms. When a timer expires to trigger a data source processing, data is collected for the physical port and for all VLL and VPLS SAPs on that port and exported using sFlow version 5 records as described in later subsections of this document. Each port and all SAP records for a data source for a specific interval are collected and sent with the counter sequence number and the timestamp value (the time value corresponds to the time counters were actually collected by a line card). The timestamp value uses line card's sysUptime value, which is synchronized with CPM time automatically by the system. A

line card sends the counters to a CPM card, where sFlow UDP datagrams are created, sequenced with the CPM sequence number and sent to the receiver. If no UDP sockets are configured, no errors are generated because data is not sent. If no UDP sockets are reachable, the created UDP sFlow datagrams are dropped.

Note: Line cards reset the counter record sequence numbers if, as a result of configuration or operational change, the return statistics no longer provide continuity with the previous interval. This may occur when:

- The card hard or soft resets.
- The MDA resets.
- The sFlow agent counter map changes.

The CPM resets the sFlow datagram sequence numbers if, as a result of configuration or operational change, the sFlow datagram to be sent no longer provides continuity with the previous datagram. The following lists examples of when this takes place:

- HA switch
- CTL reboot
- creation of an sFlow receiver

9.2.2 sFlow support on logical Ethernet ports

sFlow data sources operate in a context of physical Ethernet port. To enable sFlow on Ethernet logical ports and their SAPs, an operator must explicitly enable sFlow on every physical Ethernet port that is a member of the specific logical port. Currently only LAG logical ports are supported (including MC-LAG).

Note: sFlow configuration does not change automatically when a port is added or removed to or from a LAG.

For SAPs on a LAG, egress statistics increment based on ports used by each SAP on LAG egress while ingress statistics increment based on ports used by each SAP on LAG ingress unless LAG features like, for example, per-fp-ingress-queuing or per-fp-sap-optimization result in SAP statistics collection against a single LAG port.

If logical-level view is required, for example, per LAG statistics, a receiver is expected to perform data correlation based on per-physical port interface and SAP records exported for the logical port's physical ports and their SAPs. sFlow data records contain information that allows physical ports/SAP records correlation to a logical port. See sFlow record formats.

Note: Correlation of records must allow for small difference in timestamp values returned for member ports or SAP on a LAG because all ports run independent timestamps.

9.2.3 sFlow SAP counter map

To allow per SAP sFlow statistics export, operators must configure ingress and egress sFlow counter maps. The counter maps are required, because SR OS systems support more granular per policer/queue counters and not IF-MIB counters per VLL/VPLS SAPs. In an absence of a map configured, 0's are returned in corresponding statistics records.

A single ingress and a single egress counter map are supported. The maps specify which ingress and which egress SAP QoS policy queue/policer statistics map to sFlow unicast, multicast, and broadcast counters returned in an sFlow SAP record. Multiple queues and/or policers can map to each of unicast, multicast, broadcast counters. A single queue/policer can only map to one type of traffic. Queues, policers configured in a SAP QoS policy but not configured in an sFlow map or the other way around are ignored when sFlow statistics are collected.

9.2.4 sFlow record formats

Table 52: sFlow record fields describes sFlow record used and exported:

Table 52: sFlow record fields

Record	Field	Value
sFlow Datagram Header (SAP and port)	Datagram version	5
	Agent Address	Active CPM IPv4 address (from BoF)
	Sub-agent ID	0
	Sequence number	CPM inserted sFlow datagram sequence number
	SysUptime	SysUptime when the counters for records included in the datagram were collected by the line card
	NumSamples	Number of counter records in the datagram
Counter header (SAP and Port)	Enterprise	0 (standard sFlow)
	sFlow Sample Type	4 (Expanded counter sample)
	Sample Length	sFlow packet size excluding header
	Sequence number	Line card-inserted sequence number
	Source ID Type	0
	Source ID Index	tmnxPortId of the physical port (sFlow data source)
	Counter records	Count of counter records in the datagram
Ethernet Interface Counters (EIC) – port (Ethernet Layer)	Enterprise	Statistics returned are based on dot3Stats Entry in EtherLike-MIB.mib. Statistics support may depend on hardware type.
	Format	
	Flow data length	
	Alignment Errors	
	FCS Errors	

Record	Field	Value
	Single Collision Frames	
	Multiple Collision Frames	
	SQE Test Errors	
	Deferred Transmissions	
	Late Collisions	
	Excessive Collisions	
	Internal Mac Transmit Errors	
	Carrier Sense Errors	
	Frame Too Longs	
	Internal Mac Receive Errors	
	Symbol Errors	
Generic Interface Counters (GIC) – port/ SAP	Enterprise	0 (standard sFlow)
	Format	1 (GIC)
	Flow data length	88
	ifIndex	Port: ifIndex (tmnxPortId) of phys port SAP: SapEncapValue - part of SAP SNMP key
	ifType	Port: 6 (EthernetCsmacd) SAP: 1 (Other)
	ifSpeed	Port: Port speed value SAP: • top 32 bits: svcId for SAP (TIMETRA-SAP.mib) • lower 32 bits: sapPortId (TIMETRA-SAP.mib) The values plus ifIndex in the record are SAP SNMP key.

Record	Field	Value
		SapPortId is LAG's tmnxPortId for SAPs on a LAG and port's tmnxPortId for SAPs on physical port
	ifDirection	Derived from MAU MIB (0 = unknown, 1 = full duplex, 2 = half duplex, 3 = in, 4 = out)
	ifAdminStatus	0 (down) 1 (up)
	ifOperStatus	0 (down) 1 (up)
	Input Octets	Statistics return for port are based on ifEntry or ifXEntry in IF-MIB.mib as applicable. Statistics returned for SAPs are sum of counters based on the sFlow ingress/egress counter map configured.
	Input Packets	
	Input Multicast packets	
	Input Broadcast packets	
	Input Discarded packets	
Generic Interface Counters (GIC) – port/ SAP (Continued)	Input Errors	Statistics return for port are based on ifEntry or ifXEntry in IF-MIB.mib as applicable. Statistics returned for SAPs are sum of counters based on the sFlow ingress/egress counter map configured.
	Input Unknown Protocol Packets	
	Output Octets	
	Output Packets	
	Output Multicast packets	
	Output Broadcast packets	
	Output Discarded packets	
	Output Errors	
	Promiscuous Mode	0 (FALSE)

Note:

- 0 is returned for statistics that are not supported by a specific hardware type.
- If required, CPM executes rollover logic to convert internal 64-bit counters to a 32-bit sFlow counter returned.

10 gRPC

gRPC is a modern, open-source, high-performance Remote Procedure Call (RPC) framework that can run in any environment. In SR OS, this framework is used to implement the gRPC server, which can then be used for configuration management or telemetry.

The gRPC transport service uses HTTP/2 bidirectional streaming between the gRPC client (the data collector) and the gRPC server (the SR OS device). A gRPC session is a single connection from the gRPC client to the gRPC server over a TCP or TLS port.

The gRPC service runs on port 57400 by default in SR OS. Use the following command to configure the TCP listening port, if required.

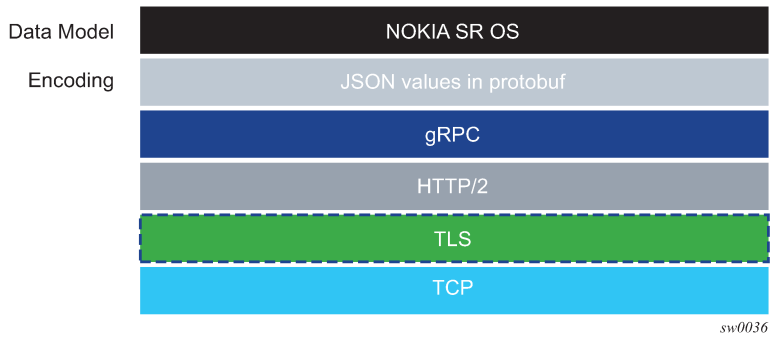
```
configure system grpc listening-port
```

A single gRPC server supports concurrent gRPC sessions and channels. The following applies:

- There is a maximum of eight concurrent gRPC sessions for all of the gRPC clients.
- There is a maximum of 225 concurrent gRPC channels for all of the gRPC clients.

The following figure shows the gRPC protocol stack.

Figure 25: Protocol stack



10.1 Security aspects

This section describes the security aspects associated with gRPC.

10.1.1 TLS-based encryption

The gRPC server on the SR OS can operate in the following modes:

- without TLS encryption
- with TLS encryption

Enabling TLS encryption provides added security. If TLS encryption is not enabled, gRPC messages are sent unencrypted; the usernames and passwords required in gRPC communication are visible if packets are captured.

Note: The following command using the **crl-optional** command option makes configuration of a valid CRL in a **ca-profile** optional. However, from a security point of view, it is important to always verify the revocation status of a certificate.

```
configure system security pki ca-profile revocation-check crl-optional
```

Before a gRPC connection comes up without TLS, the following conditions must be met:

- no TLS server profile is assigned to the gRPC server
- the following command is configured

```
configure system grpc allow-unsecure-connection
```

The following summarizes the process of encryption:

- The gRPC session must be in an encrypted state.
- If the gRPC client and server are unable to negotiate an encrypted gRPC session, the gRPC session fails and the gRPC server sends an error.
- Fallback from an encrypted to an unencrypted gRPC session is not allowed.

For information about how to configure TLS with gRPC, see the TLS chapter.

Note: SR OS TLS supports both ALPN and NPN, which are defined in RFC 7301. For any SR OS TLS server or client profile, SR OS TLS handshake always offers ALPN first, and offers NPN only if the gNMI client does not support ALPN. Consequently, no specific configurations are needed in SR OS to enable or disable ALPN or NPN extensions. For gNMI clients that use ALPN, SR OS verifies the specified HTTP2 ID and port (if needed) before replying to the gNMI client with the same HTTP2 ID and port. For gNMI clients that use NPN, SR OS retains NPN support for backward compatibility.

10.1.2 Certificate revocation check with TLS

For TLS, the following command option is supported for a Certificate Authority (CA) certificate:

```
configure system security pki ca-profile revocation-check crl-optional
```

The optional configuration can be extended for end entity (EE) certificates by using the following command (the default value is **revoked**). This command applies when the revocation status of a certificate cannot be determined because of an invalid CRL. A CRL can be invalid because it is missing, expired, or corrupted.

```
configure system security tls client-tls-profile status-verify default-result
```

Note: For safety, the configuration for EE certificates is set to **default-result revoked**. An expired CRL in the CA profile that is matched as the issuer of the EE certificate results in the EE certificate being configured as **revoked**. If the expired CRL is further up in the certificate chain, the **revocation-check crl-optional** command option works as expected.

The **status-verify** command allows users to override the recommended revocation check policy, when applicable.

See Certificate revocation check for more information.

10.1.3 Authentication

The gRPC users can be authenticated using the local user database, RADIUS, or TACACS+.

When using the local user database, the **access grpc** statement must be included in the user configuration.

- **RADIUS**
  Use the following commands to configure the **access grpc** statement and ensure that **radius use-default-template** is enabled (or the RADIUS server must send the Timetra-Access VSA with a value that includes gRPC access):
  - **MD-CLI**

    ```
    configure system security aaa user-template user-template-name radius-default
    configure system security aaa user-template access grpc true
    configure system security aaa remote-servers radius use-default-template
    ```

  - **classic CLI**

    ```
    configure system security user-template radius_default
    configure system security user-template access grpc
    configure system security radius use-default-template
    ```

- **TACACS+**
  Use the following commands to configure the **access grpc** statement and ensure that **tacplus use-default-template** is enabled:
  - **MD-CLI**

    ```
    configure system security aaa user-template user-template-name tacplus-default
    configure system security aaa user-template access grpc true
    configure system security aaa remote-servers tacplus use-default-template
    ```

  - **classic CLI**

    ```
    configure system security user-template tacplus_default
    configure system security user-template access grpc
    configure system security tacplus use-default-template
    ```

User authentication is based on following principles:

- Each RPC sent by the gRPC client carries a username and password.
- For the first RPC in the gRPC session, the gRPC server tries to authenticate the user using the specified authentication order, such as using the local user database, RADIUS, or TACACS+.

  For example, if TACACS+ is first in the authentication order, the gRPC server sends a request to the TACACS+ server to authenticate the gRPC user.
- For the subsequent RPCs on that same authenticated gRPC session, the username and password are re-authenticated only if changed.
- When no username and password are provided with the RPC, the gRPC server returns an error.

- If the RPC user is changed, any active subscriber RPCs on that same gRPC session are terminated by the gRPC server.
- If the RPC password is changed, the active gRPC session continues to exist until a different username and password is sent in a subsequent RPC, or the gRPC session is terminated.
- Each message is carried over a gRPC session that was previously encrypted; the session is not re-encrypted.

SR OS device authentication is based on the following principles:

- The gRPC clients do not share gRPC sessions. Each gRPC client starts a separate gRPC session.
- When a gRPC session is established, the gRPC server certificates are verified by the gRPC client to ensure that every gRPC server is authenticated by the gRPC client.
- If gRPC is shut down on the gRPC server and a gRPC client is trying to establish a gRPC session, the gRPC client gets an error for every RPC sent.
- If gRPC is shut down on the gRPC server and a gRPC session is established, all active RPCs are gracefully terminated and an error is returned for every active RPC.

10.2 gNMI service

The gRPC Network Management Interface (gNMI) is a gRPC based protocol for network management functions, such as changing the configuration of network elements and retrieving state information. In addition, gNMI provides functionality necessary for supporting telemetry. The gNMI service is specified in the OpenConfig forum.

10.2.1 gNMI service definitions

The SR OS gRPC server supports gNMI version 0.8.0, and, in particular, the following RPC operations:

- Capability RPC
- Get/Set RPC
- Subscribe RPC
- Publish RPC

As in NETCONF RPCs, gNMI RPCs that are sent to the SR OS system are logged in security log and they are marked as authorized or unauthorized, and include information such as username, time, RPC type, and IP address of the client.

10.2.1.1 Capability discovery

In gNMI service, the client discovers the capabilities of the gRPC server through a Capability-Discovery RPC, which consists of "CapabilityRequest" and "CapabilityResponse" messages.

During this message exchange, the gRPC server informs the client about following attributes:

- supported gNMI version
- supported models
- supported encodings

The SR OS server announces the supported models based on the configuration in the following context.

```
configure system management-interface yang-modules
```

The supported models includes the Nokia YANG or OpenConfig (OC) models.

The advertised module names and organizations are as follows:

- nokia-conf, org = "Nokia"
- nokia-state, org = "Nokia"
- openconfig, org = "OpenConfig working group" (as specified by the 'organization' in the YANG models)
- version - the version number is be defined as follows:
  - for Nokia YANG models, the version number corresponds to an SR OS release number, for example, "16.0.R1"
  - for OC YANG models, the version number corresponds to a version number defined in "oc-ext:openconfig-version" that is included in the respective YANG models
  - for OC-YANG models, including Nokia deviations, the version number corresponds to an SR OS release number, for example, "16.0.R1"

The following is an example of a "Capabilities Response Message":

```
Going to send message of type gnmi.CapabilityResponse:
 .gnmi_version: 0.4.0
 .supported_encodings (1):
  .encoding: 0 = JSON
 .supported_models (47):
  { .name: 'nokia-conf', .organization: 'Nokia', .version: '16.0.R1' }
  { .name: 'nokia-state', .organization: 'Nokia', .version: '16.0.R1' }
  { .name: 'openconfig-
bgp', .organization: 'OpenConfig working group', .version: '4.0.1' }
<snip>
  { .name: 'nokia-sr-openconfig-if-ethernet-deviations', .organization: 'Nokia',
.version: '16.0.R1' }
  { .name: 'nokia-sr-openconfig-if[[-ip-deviations', .organization: 'Nokia',
.version: '16.0.R1'..."
```

10.2.1.2 Get/Set RPC

Information is retrieved from the NE using GET RPC messages, which consist of "GetRequest" and "GetResponse" messages. The client asks for information by specifying the following:

- a set of paths

  All rules to a path definition apply, as specified in the gNMI specification.
- type

  Consists of configuration, state, or operational data.
- encoding

  This is in accordance with the server advertisement during capability discovery.
- use_models

  This message is ignored.

There is an upper limit on the size of the "GetResponse" message. This limit cannot exceed 100 MB. If the limit is exceeded, the SR OS gRPC server responds with an error message.

To modify the information in an NE element, a SET gRPC message is used. This gRPC supports three types of transactions:

- delete
- replace
- update

With a gNMI SET RPC, SR OS authorizes all configuration changes; that is, it checks the YANG tree and authorizes every changed element.

The deletion of a container results in the deletion of any children containers that are authorized for deletion, as well as their contents. Children containers that are not authorized for deletion, as well as their contents, are retained. For example, upon deletion of **configure system**, **configure system security** is not deleted because the deletion of that child container is not authorized.

Note: A "no change" for a value does not require authorization. Therefore, it is possible to execute a non-authorized command if there is no change in value.
For example, when a user is not authorized to change **access li**, but attempts to change it for another user who already has **access li**, SR OS allows that action because there is no change in value.

10.2.1.3 Subscribe RPC

The subscribe RPC is part of the telemetry support in gNMI.

The gRPC client initiates a subscription by sending a subscribe RPC that contains a "SubscribeRequest" message to the gRPC server. A prefix can be specified in the "SubscribeRequest". If a prefix is present, it is appended to the start of every path to provide a full path.

A subscription contains the following:

- a list of one or more paths with the following conditions:
  - A path represents the data tree as a series of repeated strings and elements. Each element represents a data tree node name and its associated attributes.
  - A path must be syntactically valid within the set of schema modules that the gRPC server supports.
  - The path list cannot be modified during the lifetime of the subscription.
  - If the subscription path is to a container node, all child leafs of that container node are subscribed to.
  - Any specified path must be unique within the list; paths cannot be repeated within the list. An error is returned if the same path is used more than once in a single subscription.
  - A specified path does not need to pre-exist within the current data tree on the gRPC server. If a path does not exist, the gRPC server continues to monitor for the existence of the path. Assuming that the path exists, the gRPC server transmits telemetry updates.
  - The gRPC server does not send data for a non-existent path; for example, if a path is non-existent at the time of subscription creation or if the path was deleted after the subscription was established.
  - The maximum number of paths for all subscriptions is 14400. A path using a wildcard is considered a single path.
- an encoding with the following supported options:

– JSON
– JSON-IETF
– BYTES
– PROTO

The following command affects PROTO and BYTES encoding.

```
configure system grpc gnmi proto-version
```

If the command option is **latest**, the encodings are interpreted as defined in specification v0.8.0. The following table lists the summary of different encoding options and the values used based on the **proto-version** command for all possible values defined in the gnmi.proto OpenConfig module.

Table 53: Options for the proto-version command

response field	JSON	JSON-IETF	PROTO	BYTES
string string_val = 1;			latest	v0.7.0
int64 int_val = 2;			latest	v0.7.0
uint64 uint_val = 3;			latest	v0.7.0
bool bool_val = 4;			latest	v0.7.0
bytes bytes_val = 5;			latest	v0.7.0
float float_val=6 [deprecated=true]; / / Deprecated - use double_val.				
Decimal64 decimal_val = 7 [deprecated=true]; / / Deprecated - use double_val.				v0.7.0
ScalarArray leaflist_val = 8;			latest	
google.protobuf.Any any_val = 9;				
bytes json_val = 10;	x			
bytes json_ietf_val = 11;		x		
string ascii_val = 12;				

response field	JSON	JSON-IETF	PROTO	BYTES
bytes proto_bytes = 13;			v0.7.0	latest
double double_val= 14;			latest	

- a subscription mode of one of the following types:
  - **ONCE mode**

    The server returns only one notification containing all the information to which the client has subscribed. In general, use telemetry to retrieve large amounts of information from the NE: "SubscribeRequest" message with ONCE subscription type.
  - **ON_CHANGE mode**

    The server returns notifications only when the value of the subscribed field changes. See ON_CHANGE subscription mode for more information.
  - **SAMPLE mode**

    The gRPC server sends notifications at the specified sampling interval. The first notification is sent at the next sampling interval, not immediately.
  - **TARGET_DEFINED mode**

    This type means ON_CHANGE for all states that support ON_CHANGE notifications and SAMPLE mode for all other objects in the YANG tree.
- a sample interval for each path:
  - If a sample interval of less than 1 s is specified, the gRPC server returns an error.
  - If the sample interval is set to 0, the default value of 10 s is used.
  - A sample interval is specified in nanoseconds (10 000 000 000 by default).

When a subscription is successfully initiated on the gRPC server, "SubscribeResponse" messages are sent from the gRPC server to the gRPC client. The "SubscribeResponse" message contains update notifications about the subscription path list.

An update notification contains the following:

- a timestamp of the statistics collection time, in nanoseconds
- a prefix
  - If a prefix is present, it is logically appended to the start of every path to provide the full path.
  - The presence of a prefix in the "SubscribeResponse" message is not related to the presence of a prefix in the original "SubscribeRequest" message. The prefix in the "SubscribeResponse" message is optimized by the gRPC server.
- a list of updates (path and value pairs)
  - A path represents the data tree path as a series of repeated strings or elements, where each element represents a data tree node name and its associated attributes. See Schema paths for more information.
  - The "TypedValue" message represents the value of the data tree node, where the encoding is "JSON", "JSON_IETF", "Bytes, or "Protobuf" depending on the information in the "SubscribeRequest" message.

Multiple notification messages can be combined in a single "SubscribeResponse" message. This bundling minimizes overhead, which improves the efficiency of telemetry data transport, however, this may delay some notifications and timestamps may be less accurate. The following commands control message bundling:

- **max-time-granularity**

  This parameter controls the maximum time during which notifications can be bundled.

- **max-msg-count**

  This parameter controls the maximum number of notifications that can be bundled.

A synchronization response notification is sent once, after the gRPC server sends all updates for the subscribed-to paths. The synchronization response must be set to "true" for the gRPC client to consider that the stream has synced one time. A sync response is used to signal the gRPC client that it has a full view of the subscribed-to data.

The gRPC server sends an error, if required. The error contains a description of the problem.

Authorization checks are not performed by default for telemetry data. All configuration and state elements are available to authenticated telemetry subscriptions, with the exception of LI (Lawful Intercept) configuration and state elements, which are authorized separately based on the LI authorization configuration. To control telemetry data authorization, use the following command:

- **MD-CLI**

```
configure system security aaa management-interface output-authorization telemetry-data
```

- **classic CLI**

```
configure system security management-interface output-authorization telemetry-data
```

10.2.1.3.1 PROTO encoding

PROTO encoding is performed by the gRPC server, which encodes the values of the leafs as typed values. The following table lists the mapping of the individual YANG types to the typed values defined in the gNMI specification (v0.8.0) in the GitHub repository.

Table 54: Mapping of YANG types to gNMI-specified typed values

YANG type	Typed value
binary	bytes_val
bits	string_val
boolean	bool_val
decimal64	double_val
empty	bool_val
enumeration	string_val
identityref	string_val

YANG type	Typed value
instance-identifier	string_val
int8	int_val
int16	int_val
int32	int_val
int64	int_val
leafref	type of referenced leaf
string	string_val
uint8	uint_val
uint16	uint_val
uint32	uint_val
uint64	uint_val
union	type of active union member

10.2.1.3.2 BYTES encoding

The BYTES encoding mechanism uses the binary encoding format for both path and value to increase the efficiency of telemetry data transfer. The YANG files are encoded in protobuf format, in which every possible path is encoded as a binary index.

The protobuf encoded YANG files are distributed together with the SR OS software or found on https://github.com under Nokia 7X50_protobufs. The PROTO definitions of the YANG model are specifically generated for each software release; backward compatibility is not supported.

To retrieve decoding information for PROTO encoding, the gNMI client can use GetRPC to request a path with origin "gnmi.schemas". The path must be either a root path "gnmi.schemas:/", or a specific path, for example, "gnmi.schemas:/protobuf-typemap", as described in *protobuf-vals.md* published on github.com (May 2018).

Example

The following example shows the GetRPC request and SR OS response:

```
Received message of type gnmi.GetRequest:
.type: 0 = ALL
.encoding: 4 = JSON_IETF
.prefix: /
.path (1):
  .path: gnmi.schemas:/
"
Going to send message of type gnmi.GetResponse:
.notification (1):
  .timestamp: 1616677860728392525
  .update (1):
   .path: gnmi.schemas:/protobuf-typemap
```

```
.val.value = json_ietf_val: {
 "gnmi-protobuf-encoding:origin": [
     {
         "name": "",
         "container": [
             {
                 "path": "/",
                 "message-name": "proto.nokia.net/Nokia.SROS.root"
             }
         ]
     }
```

10.2.1.3.3 ON_CHANGE subscription mode

The SR OS supports ON_CHANGE subscription mode. This subscription mode indicates that notification messages are sent as follows:

- after the "SubscriptionRequest" message is received
- every time the corresponding leaf value is changed

The notification message, as a response to an ON_CHANGE subscription, always contains the new value of the corresponding leaf, as defined in the gNMI specification.

The ON_CHANGE subscription is supported for all configuration leafs as well as for selected state leafs. Use the following command to display all state leafs supporting the ON_CHANGE subscription.

```
tools dump system telemetry on-change-paths
```

ON_CHANGE subscription is accepted for all valid paths. The server sends ON_CHANGE notifications only for leafs within the specified subtree that support ON_CHANGE notifications.

10.2.1.4 Publish RPC

With dial-out telemetry, where the SR OS node is the gRPC client instead of the gRPC server, the SR OS node sends a Publish RPC with a "SubscribeResponse" message to the gRPC server. See Dial-out telemetry for more information.

Because the current gnmi.proto definition does not support dial-out mode, a protobuf definition is introduced, with a separate gRPC service, as follows:

```
NOKIA-DialOut.proto
      option (dialout_service) = 0.1.0
      service gMIDialOut {
              rpc Publish(stream SubscribeResponse) returns (stream PublishResponse)
      }
      message PublishResponse {
      }
```

The preceding proto file definition reuses the "SubscribeResponse" message defined for dial-in telemetry, in accordance with the gNMI specification.

10.2.1.5 Schema paths

Telemetry subscriptions include a set of schema paths to identify which data nodes are of interest to the collector.

The paths in Telemetry Subscribe RPC requests follow the basic conventions described in the *OpenConfig gnmi-path-conventions.md* published on github.com (version 0.4.0, published June 21, 2017).

A path consists of a set of path segments, often shown with a "/" character as a delimiter.

Example

```
/configure/router[router-name=Base]/interface[interface-name=my-interface1]/description
```

These paths are encoded as a set of individual string segments in gnmi.proto (without any "/" characters).

Example

```
["configure", "router[router-name=Base]", "interface[interface-name=my-interface1]",
 "description"]
```

A path selects an entire subtree of the data model and includes all descendants of the node indicated in the path. The following table describes the types of schema paths that are supported in SR OS telemetry.

Table 55: Schema paths

Path example	Description
/configure/router[router-name=Base]/ interface[interface-name=abc]	Selects all config leafs of interface abc and all descendants
/configure/router[router-name=Base]/ interface[interface-name=abc]/description	Selects only the description leaf of interface abc
/state/router[router-name=Base]/ interface[interface-name=*]	Selects all state information for all base router interfaces using a wildcard in a single segment of a path
/configure/router[router-name=Base]/ interface[interface-name=*]/description	Selects all state information for all base router interfaces using a wildcard in a single segment of a path
/configure/router[router-name=Base]/ interface/description	Applies to all RPCs. Selects the entire list to account for absent list keys. It is equivalent to /configure/router[router-name=Base]/ interface[interface-name=*]/description
/	The root path. This selects all config and state data from all models (in all namespaces) supported on the router. Encoded as "" in gRPC/gPB.
/state/...	Not supported. Equivalent to /state/

Path example	Description
/*	Expands one level of a specified subtree schema.

The following list describes telemetry paths that are supported in SR OS:

- Wildcards in multiple segments of a path are supported.

  For example: /state/card[slot-number=*]/mda[mda-slot=*]
- The following wildcards are supported in the schema.
  - Specifying "/…" wildcard expands to multiple element levels in a path.
  - Specifying "/*" wildcard expands to only one level in a path.
- Wildcards for entire path segments are supported as follows:
  - For example: "/state/card/.../oper-state" expands to following paths

    /state/card[slot-number=*]/hardware-data/oper-state

    /state/card[slot-number=*]/mda[mda-slot=*]/hardware-data/oper-state

    /state/card[slot-number=*]/mda[mda-slot=*]/flex[group-index=*]/oper-state
  - For example: "/state/card/*/oper-state" expands to following path

    /state/card[slot-number=*]/hardware-data/oper-state
- The paths with wildcards are expanded when a subscription is activated; this applies to dynamic and persistent subscriptions. In some cases, it is possible that a single path with wildcards can be expanded across both Nokia and Openconfig YANG models. However, this occurs only if both model types are enabled. If only one type is enabled, the path is expanded only within the enabled model. If the other type is enabled later, it is necessary to reset all subscriptions, which ensures that the expansion includes the newly enabled model type.

The following list describes telemetry paths that are not supported in SR OS.

- If a wildcard is used for any key of a list, a wildcard must be used for all the keys of that list. In a single path segment, all keys must either have specific values, or all keys must have wildcards. A mix of wildcards and specific values for different parts of a list key is not supported.

  For example:

  Supported:

  /a/b[key1=*][key2=*]/c[key1=foo]

  /a/b[key1=foo][key2=bar]/c[key1=*]

  Not supported:

  /a/b[key1=foo][key2=*]
- Functions such as "current()" and "last()", and mathematical operators, such as stat<5 or octets>3 are not supported in paths. The "|" (OR operator, used to select multiple paths) is not supported.

10.2.2 gNMI service use cases

The gNMI service can be used for the following:

- Telemetry
- NE Configuration Management

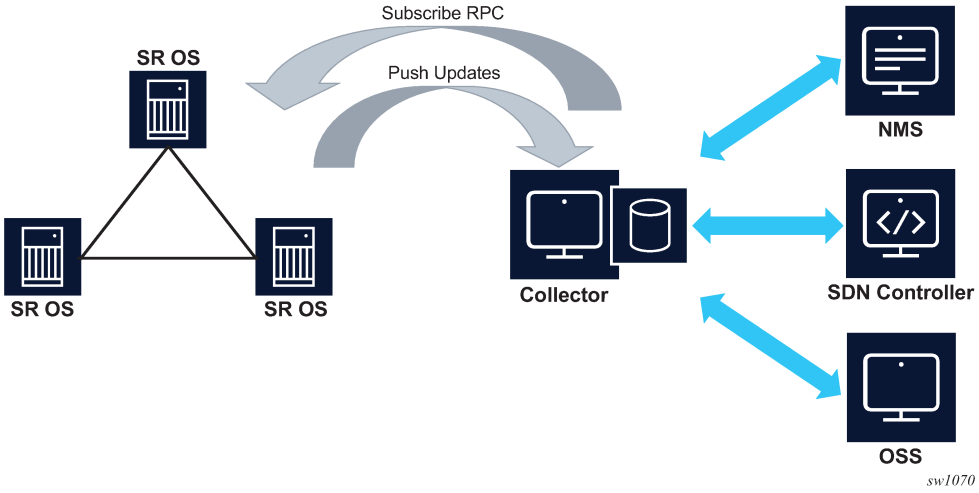
10.2.2.1 Telemetry

Telemetry is a network monitoring and fault management framework. Telemetry is driven by the need to use fresh data obtained from the network to make fast networking decisions, such as traffic optimization and preventive troubleshooting.

10.2.2.1.1 Dial-in telemetry

When the data collector initiates the gRPC connection, the SR OS node assumes the role of the gRPC server and the collector is the client. This is referred to as dial-in telemetry, where the SR OS node pushes data to the receiver (collector). The following figure shows the telemetry session initiated from the collector to the SR OS node via the Subscribe RPC.

Figure 26: Dial-in telemetry session



Authorization checks are not performed by default for telemetry data. All configuration and state elements are available to authenticated telemetry subscriptions, with the exception of LI (Lawful Intercept) configuration and state elements, which are authorized separately based on the LI authorization configuration. Use the following command to control telemetry data authorization:

- **MD-CLI**

```
configure system security aaa management-interface output-authorization telemetry-data
```

- **classic CLI**

```
configure system security management-interface output-authorization telemetry-data
```

10.2.2.1.1.1 Dynamic subscriptions

Dynamic subscriptions are created by the collector using the Subscribe RPC. These subscriptions are removed as soon as the gRPC session terminates. Dynamic subscriptions are currently supported only in dial-in mode.

10.2.2.1.1.2 Dial-in telemetry examples

This section contains examples of telemetry subscription requests and responses. The following examples are dumps of protobuf messages from a Python API. Formats may vary across different implementations.

Example: Subscribe to a single path

```
2017-06-05 17:06:13,189 - SENT::SubscribeRequest
subscribe {
  subscription {
    path {
      element: "state"
      element: "router[router-instance=Base]"
      element: "interface[interface-name=test]"
      element: "statistics"
      element: "ip"
      element: "in-packets"
    }
    mode: SAMPLE
    sample_interval: 10000000000
  }
}
```

```
2017-06-05 17:06:13,190 - RCVD::SubsribeResponse
2017-06-05 17:06:23,492 - RCVD::Subscribe
2017-06-05 17:06:23,492 - update {
  timestamp: 1496675183491595139
  prefix {
    element: "state"
    element: "router[router-instance=Base]"
    element: "interface[interface-name=test]"
    element: "statistics"
    element: "ip"
  }
  update {
    path {
      element: "in-packets"
    }
    val {
      json_val: ""0""
    }
  }
}
2017-06-05 17:06:23,494 - RCVD::Subscribe
2017-06-05 17:06:23,494 - sync_response: true
```

```
2017-06-05 17:06:33,589 - RCVD::Subscribe
2017-06-05 17:06:33,589 - update {
  timestamp: 1496675213491595139
  prefix {
    element: "state"
```

```
    element: "router[router-instance=Base]"
    element: "interface[interface-name=test]"
    element: "statistics"
    element: "ip"
  }
  update {
    path {
      element: "in-packets"
    }
    val {
      json_val: ""28""
  }
}
....
....
```

Example: Subscribe to a single path with wildcard

```
2017-06-05 17:08:29,055 - SENT::SubscribeRequest
subscribe {
  subscription {
    path {
      element: "state"
      element: "router[router-instance=Base]"
      element: "interface[interface-name=*]"
      element: "statistics"
      element: "ip"
      element: "in-packets"
    }
    mode: SAMPLE
    sample_interval: 30000000000
  }
}
```

```
2017-06-05 17:08:29,056 - RCVD::SubsribeResponse
2017-06-05 17:08:59,133 - RCVD::Subscribe
2017-06-05 17:08:59,133 - update {
  timestamp: 1496675339132056575
  prefix {
    element: "state"
    element: "router[router-instance=Base]"
    element: "interface[interface-name=system]"
    element: "statistics"
    element: "ip"
  }
  update {
    path {
      element: "in-packets"
    }
    val {
      json_val: ""0""
    }
  }
}
2017-06-05 17:08:59,135 - RCVD::Subscribe
2017-06-05 17:08:59,135 - update {
  timestamp: 1496675339133006678
  prefix {
    element: "state"
    element: "router[router-instance=Base]"
    element: "interface[interface-name=to_node_B]"
    element: "statistics"
```

```
    element: "ip"
  }
  update {
    path {
      element: "in-packets"
    }
    val {
      json_val: ""0""
    }
  }
}
2017-06-05 17:08:59,135 - RCVD::Subscribe
2017-06-05 17:08:59,135 - update {
  timestamp: 1496675339133006678
  prefix {
    element: "state"
    element: "router[router-instance=Base]"
    element: "interface[interface-name=to_node_D]"
    element: "statistics"
    element: "ip"
  }
  update {
    path {
      element: "in-packets"
    }
    val {
      json_val: ""0""
    }
  }
}
2017-06-05 17:08:59,136 - RCVD::Subscribe
2017-06-05 17:08:59,136 - sync_response: true
```

```
2017-06-0517:09:29,139 - RCVD::Subscribe
2017-06-0517:09:29,139 - update {
  timestamp: 1496682569121314
  prefix {
    element: "state"
    element: "router[router-instance=Base]"
    element: "interface[interface-name=system]"
    element: "statistics"
    element: "ip"
  }
  update {
    path {
      element: "in-packets"
    }
    val {
      json_val: ""0""
    }
  }
}
2017-06-05 17:09:29,142 - RCVD::Subscribe
2017-06-05 17:09:29,142 - update {
  timestamp: 1496682569124342
  prefix {
    element: "state"
    element: "router[router-instance=Base]"
    element: "interface[interface-name=to_node_B]"
    element: "statistics"
    element: "ip"
  }
  update {
```

```
    path {
      element: "in-packets"
    }
    val {
      json_val: ""0""
    }
  }
}
2017-06-05 17:09:29,145 - RCVD::Subscribe
2017-06-05 17:09:29,145 - update {
  timestamp: 1496682569127344
  prefix {
    element: "state"
    element: "router[router-instance=Base]"
    element: "interface[interface-name=to_node_D]"
    element: "statistics"
    element: "ip"
  }
  update {
    path {
      element: "in-packets"
    }
    val {
      json_val: ""0""
    }
  }
}
....
....
```

Example: Subscribe to more than one path

```
2017-01-24 12:54:18,228 - SENT::SubscribeRequest
subscribe {
  subscription {
    path {
      element: "state"
      element: "router[router-instance=Base]"
      element: "interface[interface-name=to_node_B]"
    }
    mode: SAMPLE
    sample_interval: 30000000000
  }
  subscription {
    path {
      element: "state"
      element: "router[router-instance=Base]"
      element: "mpls"
      element: "statistics"
      element: "lsp-egress-stats[lsp-name=lsp_to_dest_f]"
    }
    mode: SAMPLE
    sample_interval: 30000000000
  }
}
```

Example: Subscribe to a list with wildcard

```
2017-01-24 13:45:30,947 - SENT::SubscribeRequest
subscribe {
  subscription {
    path {
```

```
      element: "state"
      element: "router[router-instance=Base]"
      element: "interface[interface-name=*]"
    }
    mode: SAMPLE
    sample_interval: 30000000000
  }
}
```

Example: Subscribe to path where the object did not exist before subscription

```
2017-01-24 13:53:50,165 - SENT::SubscribeRequest
subscribe {
  subscription {
    path {
      element: "state"
      element: "router[router-instance=Base]"
      element: "interface[interface-name=to_node_B]"
    }
    mode: SAMPLE
    sample_interval: 30000000000
  }
}
```

```
2017-01-24 13:53:50,166 - RCVD::SubsribeResponse
2017-01-24 13:54:20,169 - RCVD::Subscribe
2017-01-24 13:54:20,169 - sync_response: true
```

```
2017-01-24 13:54:50,174 - RCVD::Subscribe
2017-01-24 13:54:50,174 - update {
  timestamp: 1485262490169309451
  prefix {
    element: "state"
    element: "router[router-instance=Base]"
    element: "interface[interface-name=to_node_B]"
  }
  update {
...
...
  }
}
```

Example: Subscribe to a path where the object existed before subscription and was then deleted after subscription

```
2017-01-24 14:00:41,292 - SENT::SubscribeRequest
subscribe {
  subscription {
    path {
      element: "state"
      element: "router[router-instance=Base]"
      element: "interface[interface-name=to_node_B]"
    }
    mode: SAMPLE
    sample_interval: 30000000000
  }
}
```

```
2017-01-24 14:00:41,294 - RCVD::SubsribeResponse
```

```
2017-01-24 14:01:11,295 - RCVD::Subscribe
2017-01-24 14:01:11,295 - update {
  timestamp: 1485262871290064704
  prefix {
    element: "state"
    element: "router[router-instance=Base]"
    element: "interface[interface-name=to_node_B]"
  }
  update {
...
...
  }
}
2017-01-24 14:01:11,359 - RCVD::Subscribe
2017-01-24 14:01:11,359 - sync_response: true
```

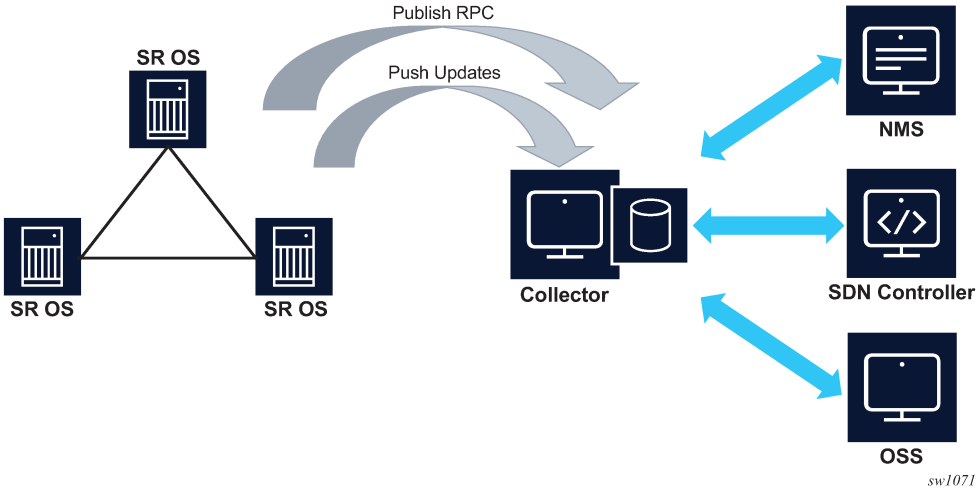
```
2017-01-24 14:01:41,293 - RCVD::Subscribe
```

```
2017-01-24 14:02:11,296 - RCVD::Subscribe
```

10.2.2.1.2 Dial-out telemetry

When the SR OS node initiates the gRPC connection, the SR OS node assumes the role of the gRPC client. This is referred to as dial-out telemetry. The following figure shows the telemetry session initiated from the SR OS node to the collector via a Publish RPC.

Figure 27: Dial-out telemetry session



10.2.2.1.2.1 Persistent subscriptions

Persistent subscriptions are configured on the SR OS node and they are not cleared when the gRPC session terminates. Persistent subscriptions are supported only in dial-out mode.

A persistent subscription associates one or more paths with corresponding destinations via sensor groups and a local user for data authorization.

Every subscription has an associated administrative state as well as an operational state. If a connection is lost, the operational state goes down. If the collector does not receive the data but the SR OS appears to have a connection and the subscription is up, the connection can be reset by setting the administrative state down and then back up.

Destinations are defined in the form of destination groups. A destination group supports up to four destinations, where the destinations are served in a round-robin fashion. SR OS attempts to connect the first destination, and if successful, the telemetry data is sent to that destination. If the connection to the first destination fails (initially or during operation), SR OS attempts to connect or reconnect to the second destination, if it is configured. All configured destinations and local addresses should be reachable in the specified routing instances.

When the SR OS node initiates the gRPC connection via the Publish RPC, it includes the subscription name and the configured system name in the metadata. The collector can use this information to associate individual notification messages with the node and subscription.

Modifying any parameter of the active subscription causes the SR OS node to close the gRPC connection before attempting a reconnection.

When a gRPC connection is lost, the SR OS node continually attempts to establish a new session with the collector.

10.2.2.1.2.2 QoS marking

The QoS marking of the IP packets carrying notifications can be configured under persistent subscription. IP packets to a specified destination are marked according to the configuration of the first subscription opened to the destination. This DSCP marking is maintained, regardless of any configuration changes, as long as the dial-out connection to the specified destination is open. If the destination is disconnected for any reason, the DSCP marking must be redefined when the connection is reestablished.

10.2.2.1.2.3 Configuring dial-out telemetry

The dial-out telemetry configuration process includes the following elements:

- **telemetry-default-user**
  The telemetry default user specifies the local user for output authorization. This local user must have the following permissions:
  - Access to gRPC enabled in the user profile
  - A local command authorization profile with **match** and **action accept** entries to control access to specific subscription paths, or a **default-action** of **permit-all** if access to all paths should be permitted
- **sensor group**

  The sensor group specifies one or more schema paths in XPath format from which data is streamed to the collector.
- **destination group**

  The destination group specifies the destination addresses (and ports) that the router uses to send the telemetry data.
- **persistent subscription**

The persistent subscription associates a sensor group with a destination group and specifies streaming parameters for the telemetry data. For example, the subscription mode can be specified (ON_CHANGE, SAMPLE, or TARGET_DEFINED) for the subscription.

Dial-out telemetry can be configured via the MD-CLI or the classic CLI. For more information about using the MD-CLI, see the *7705 SAR Gen 2 MD-CLI User Guide*. For more information about the MD-CLI configuration commands, see the *7705 SAR Gen 2 MD-CLI Command Reference Guide*.

For more information about the classic CLI configuration commands, see the *7705 SAR Gen 2 Classic CLI Command Reference Guide* and the *7705 SAR Gen 2 Clear, Monitor, Show, Tools CLI Command Reference Guide*.

The following example shows a dial-out telemetry configuration. The **path** must be enclosed in quotation marks (") when it includes a list key, for example `"/state/router[router-name=Base]"`.

Example: MD-CLI

```
[ex:/configure system security user-params local-user user "grpc"]
A:admin@node-2# info
    password "$2y$10$f7ZHZbD6BSu/CwX0a8eeo.i3p1NkPiIyHS8c/96j12bg2kce2yle2"
    access {
        grpc true
    }
    console {
        member ["grpc"]
    }

[ex:/configure system security aaa local-profiles profile "grpc"]
A:admin@node-2# info
    default-action deny-all
    entry 5 {
        match "state router interface statistics ip"
        action permit
    }

[ex:/configure system security aaa management-interface output-authorization]
A:admin@node-2# info
    telemetry-default-user "grpc"

[ex:/configure system telemetry]
A:admin@node-2# info
    destination-group "quick_cfg_dg_1" {
        description "Destination Group 1"
        allow-unsecure-connection
        destination 192.168.65.5 port 40001 {
            router-instance "Base"
        }
        destination 192.168.65.5 port 40002 {
            router-instance "Base"
        }
    }
    persistent-subscriptions {
        subscription "quick_cfg_sub_1" {
            admin-state enable
            description "Subscription 1"
            sensor-group "quick_cfg_sg"
            mode sample
            sample-interval 1234
            destination-group "quick_cfg_dg_1"
            local-source-address 192.168.3.4
            originated-qos-marking cp19
            encoding bytes
```

```
        }
    }
    sensor-groups {
        sensor-group "quick_cfg_sg" {
            description "Sensor Group"
            path "/state/router[router-name=Base]/interface[interface-name=test]/
statistics/ip" {
            }
        }
    }
```

Example: classic CLI

```
A:node-2>config>system>security>user# info
----------------------------------------------
                password "$2y$10$f7ZHZbD6BSu/CwX0a8eeo.i3p1NkPiIyHS8c/96j12bg2kce2yle2"
                access grpc
                console
                    no member "default"
                    member "grpc"
                exit
----------------------------------------------

A:node-2>config>system>security>profile# info
----------------------------------------------
                default-action deny-all
                entry 5
                    match "state router interface statistics ip"
                    action permit
                exit
----------------------------------------------

A:node-2>config>system>security>management-interface>output-authorization# info
----------------------------------------------
                    telemetry-default-user "grpc"
----------------------------------------------

A:node-2>config>system>telemetry# info
----------------------------------------------
            destination-group "quick_cfg_dg_1" create
                description "Destination Group 1"
                allow-unsecure-connection
                tcp-keepalive
                    shutdown
                exit
                destination 192.168.65.6 port 40001 create
                    router-instance "Base"
                exit
                destination 192.168.65.5 port 40002 create
                    router-instance "Base"
                exit
            exit
            sensor-groups
                sensor-group "quick_cfg_sg" create
                    description "Sensor Group"
                    path "/state/router[router-name=Base]/interface[interface-name=test]/
statistics/ip" create
                    exit
                exit
            exit
            persistent-subscriptions
                subscription "quick_cfg_sub_1" create
                    description "Subscription 1"
```

```
                    destination-group "quick_cfg_dg_1"
                    encoding bytes
                    mode sample
                    sample-interval 1234
                    sensor-group "quick_cfg_sg"
                    local-source-address 192.168.3.4
                    originated-qos-marking "cp19"
                    no shutdown
                exit
            exit
-----------------------------------------
```

Use the following command to show the telemetry persistent subscription for "quick_cfg_sub_2".

```
show system telemetry persistent subscription "quick_cfg_sub_2"
```

Output example

```
===============================================================================
Telemetry persistent subscription
===============================================================================
Subscription Name     : quick_cfg_sub_2
Administrative State  : Enabled
Operational State     : Up
Subscription Id       : 198
Description           :
Sensor Group          : quick_cfg_sg_2
Destination Group     : quick_cfg_dg_2
Path Mode             : sample
Sample Interval       : 1000 ms
Encoding              : bytes
===============================================================================
```

Use the following command to show the telemetry persistent subscription paths for "quick_cfg_sub_2".

```
show system telemetry persistent subscription "quick_cfg_sub_2" paths
```

Output example

```
===============================================================================
Telemetry persistent subscriptionrsistent# subscription "quick_cfg_sub_2" paths
===============================================================================
Subscription Name     : quick_cfg_sub_1nt# subscription "quick_cfg_sub_2" paths
Administrative State  : Enabled
Operational State     : Up
Subscription Id       : 198
Description           :
Sensor Group          : quick_cfg_sg_2
Destination Group     : quick_cfg_dg_2
Path Mode             : sample
Sample Interval       : 1000 ms
Encoding              : bytes
-------------------------------------------------------------------------------
Paths
-------------------------------------------------------------------------------
Path                  : /state
Finished Samples      : 178
Deferred Samples      : 402
Total Collection Time : 405223 ms
Min Collection Time   : 2021 ms
Avg Collection Time   : 2276 ms
```

```
Max Collection Time   : 2956 ms
-------------------------------------------------------------------------------
No. of paths          : 1
===============================================================================
```

Use the following command to show the telemetry persistent subscription destinations for "quick_cfg_sub_2".

```
show system telemetry persistent subscription "quick_cfg_sub_2" destinations
```

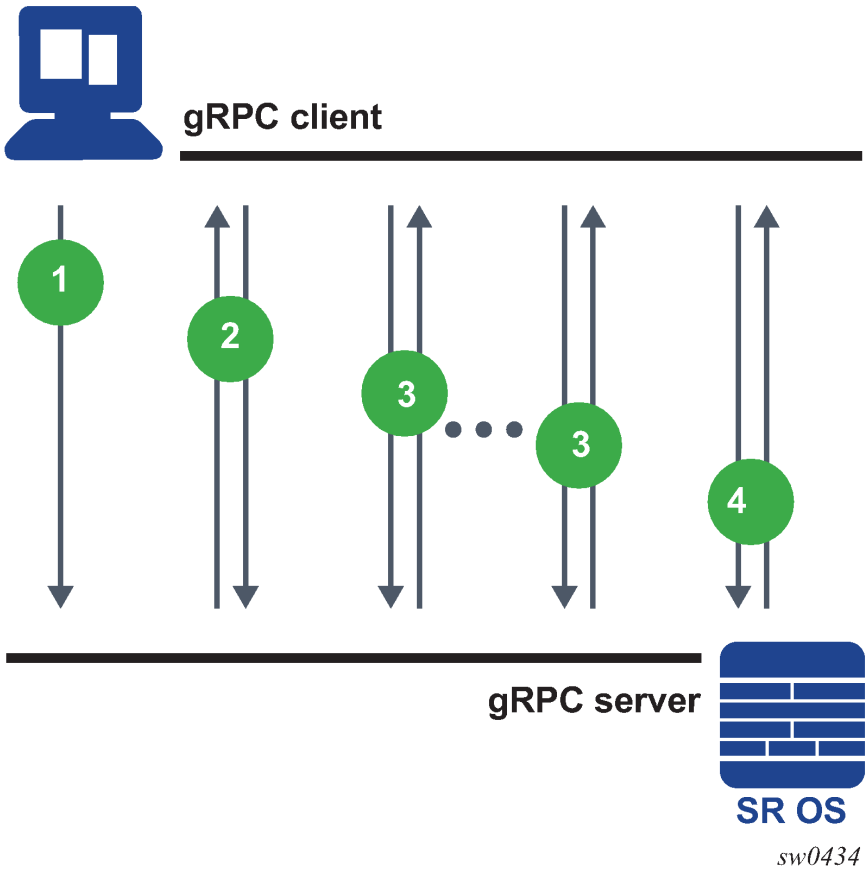
Output example

```
===============================================================================
Telemetry persistent subscription
===============================================================================
Subscription Name     : quick_cfg_sub_2
Administrative State  : Enabled
Operational State     : Up
Subscription Id       : 198
Description           :
Sensor Group          : quick_cfg_sg_2
Destination Group     : quick_cfg_dg_2
Path Mode             : sample
Sample Interval       : 1000 ms
Encoding              : bytes
-------------------------------------------------------------------------------
Destinations
-------------------------------------------------------------------------------
Destination           : 192.168.65.1
Port                  : 40001
Operational State     : Down
Last Oper Down Reason : MINOR: TELEMETRY #2353: RPC refused by peer
Last Oper Change      : 2020/04/06 21:19:29
Connection Attempts   : 22
Notification Count    : 0
Total Notification Co*: 2315653
-------------------------------------------------------------------------------
Destination           : 192.168.65.1
Port                  : 40002
Operational State     : Up
Last Oper Down Reason : MINOR: TELEMETRY #2356: Canceled by config change
Oper Router Instance  : management
Last Oper Change      : 2020/04/06 21:19:30
Connection Attempts   : 22
Notification Count    : 3783151
Total Notification Co*: 5034573
-------------------------------------------------------------------------------
No. of destinations   : 2
* indicates that the corresponding row element may have been truncated.
===============================================================================
```

10.2.2.2 Configuration management via NMS

The following figure shows NMS configuration and information retrieval using the gNMI service.

Figure 28: NE configuration and information retrieval using gNMI service



In the context of gNMI, every Set RPC appears as an single commit operation, regardless of the number of paths included in the message. Nokia and OpenConfig models are supported by gNMI Get or Set RPCs.

The following are examples of the Get RPC and Set RPC commands, including the response messages from the gRPC server.

Example: Get RPC command

```
gNMI_rpc - INFO - SENT::GetRequest GET140550212650064
path {
  elem {
    name: "configure"
  }
  elem {
    name: "system"
  }
  elem {
    name: "location"
  }
}
type: CONFIG
2017-12-06 12:17:28,639 - gMI_rpc - INFO -
 RCVD::GetResponse GET140550212650064
notification {
  timestamp: 1512559048634751055
```

```
  update {
  path {
    elem {
      name: "configure"
    }
    elem {
      name: "system"
    }
    elem {
      name: "location"
    }
  }
  val {
    json_val: "zurich"
  }
 }
}
```

Example: Set RPC command

```
 gNMI_rpc - DEBUG - SENT::SetRequest
prefix {
}
update {
  path {
    elem {
      name: "configure"
    }
    elem {
      name: "system"
    }
 }
 val {
   json_val: {"location": "zurich"}
 }
}
gMI_rpc - DEBUG - RCVD::SetResponse
prefix {
}
response {
  path {
    elem {
      name: "configure"
    }
    elem {
      name: "system"
    }
  }
  op: UPDATE
}
```

10.3 gNOI services

The gRPC Network Operations Interface (gNOI) defines a set of gRPC-based micro-services for executing operational commands on network devices. This includes the gNOI CERT service, which provides certificate management. The individual RPCs and messages that perform the operations required for certificate management on the node are defined in the Git repository hosting service (GitHub).

10.3.1 Certificate management for TLS connections

This section describes the gNOI services certificates for managing secure Transport Layer Security (TLS) connections.

The SR OS supports the following RPCs for managing certificates for secure TLS connections:

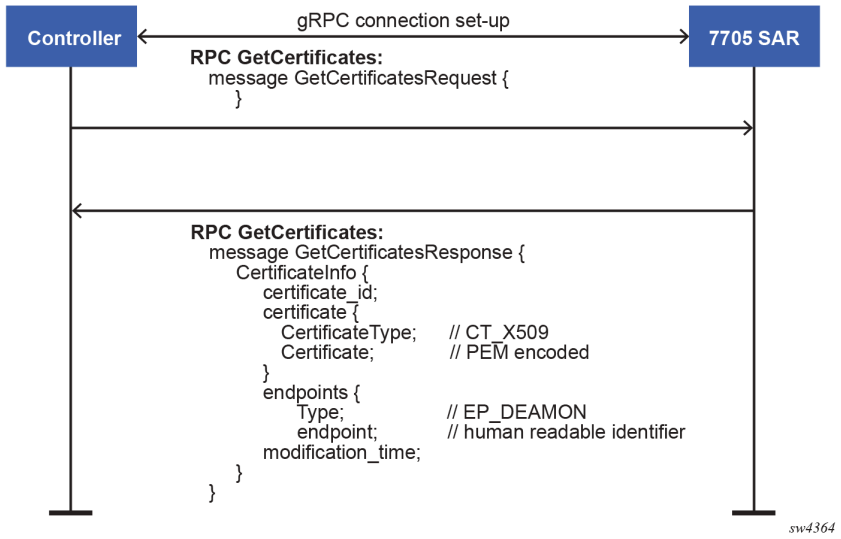
- RPC GetCertificates
- RPC CanGenerateCSR
- RPC Rotate
- RPC Install
- RPC RevokeCertificates

Note: By default, gNOI RPCs are disabled in the user profile.

10.3.1.1 RPC GetCertificates

RPC GetCertificates provide information to the controller about all active certificates on the server (SR OS node). The following figure shows the message flow.

Figure 29: RPC GetCertificates message flow



The RPC GetCertificates messages include a GetCertificateRequest and a GetCertificateResponse message. The GetCertificatesResponse message shown in the preceding figure includes the following information:

- **certificate_id**

  The SR OS uses a certificate filename as the certificate ID.

- **CertificateType**

This is always set to X509 because it is the only type that the SR OS supports.

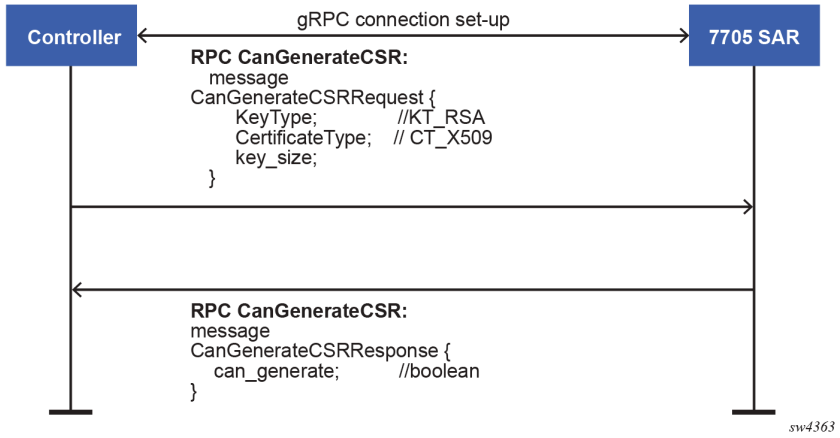
- **endpoint**

  Indicates the CERT profiles in the SR OS node that use this certificate; if multiple CERT profiles use the certificate, the names are concatenated with the separation character "/".

10.3.1.2 RPC CanGenerateCSR

The RPC CanGenerateCSR message can be used to determine whether the gRPC server (SR OS node) can generate a Certificate Signing Request (CSR). It is a simple request and response operation, as shown in the following figure.

Figure 30: RPC CanGenerateCSR message flow



The SR OS supports only RSA keys and X509 certificates, so it responds positively only when those values are present in the respective fields. The key size must be between 512 and 8192. In all other cases, the SR OS responds negatively to the CanGenerateCSRRequest message.

10.3.1.3 RPC Rotate

RPC Rotate allows the controller to rotate an active certificate on the server. After the rotation is completed, a new certificate can be used without affecting existing TLS connections.

The following cases are supported for a certificate rotation:

- server capable of generating a CSR (see Figure 31: RPC Rotate message flow for CSRs generated on the SR OS node)
- server not capable of generating a CSR (see Figure 32: RPC Rotate message flow when CSRs are not generated on the SR OS node)

The SR OS supports both scenarios, although it is assumed that in most cases the CSR is generated on SR OS node.

The following steps apply to both scenarios:

1. Generate the CSR.

2. Sign the CSR by the Certificate Authority (CA).
3. Load the new certificate on the server.
4. Verify the new certificate by creating a new connection.
5. Finalize by confirming that the new certificate is being used.

After the RPC Rotate is completed, all new connections use new keys.

Figure 31: RPC Rotate message flow for CSRs generated on the SR OS node

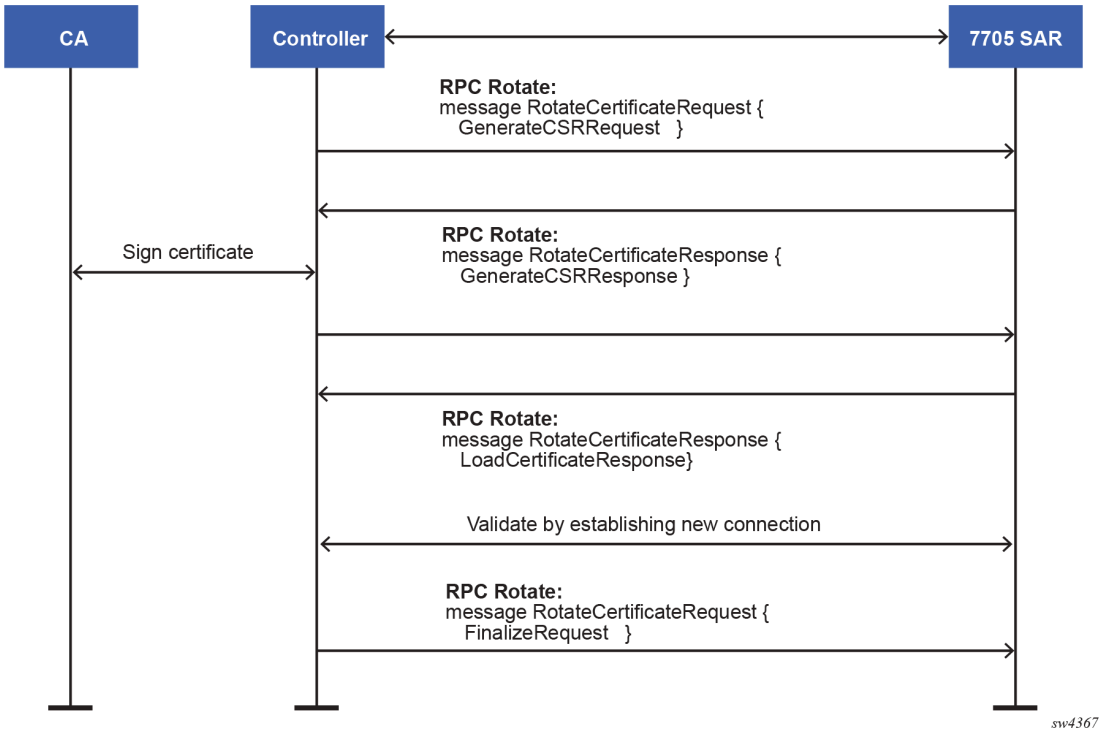
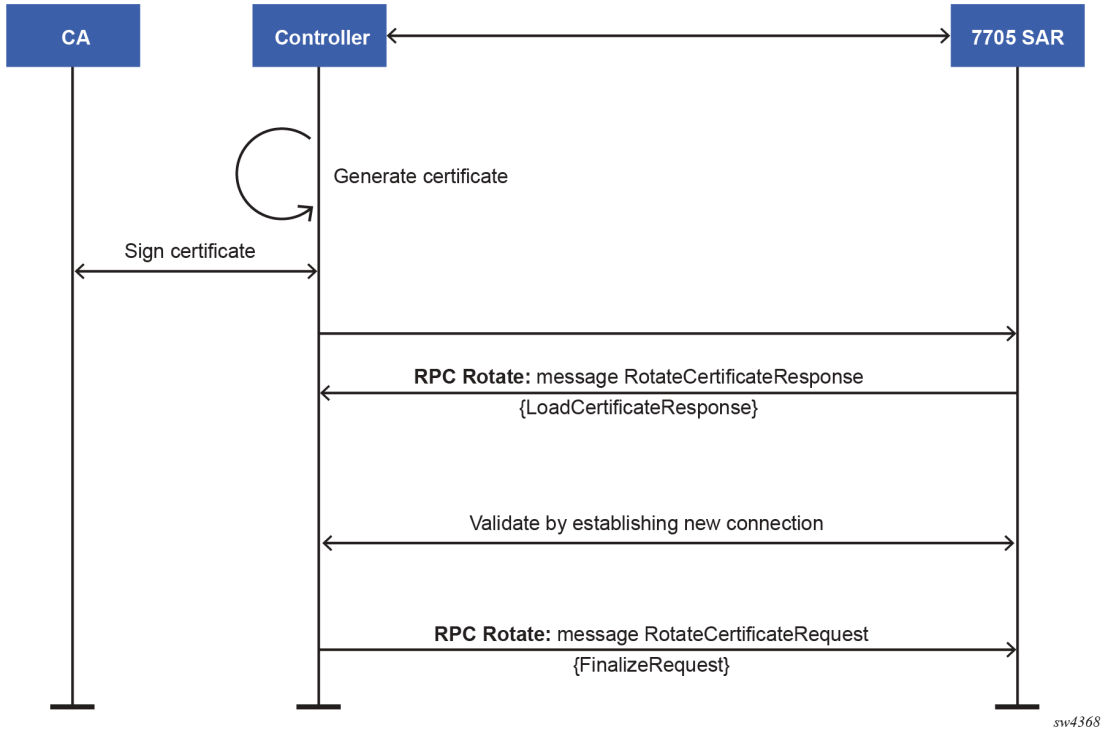


Figure 32: RPC Rotate message flow when CSRs are not generated on the SR OS node

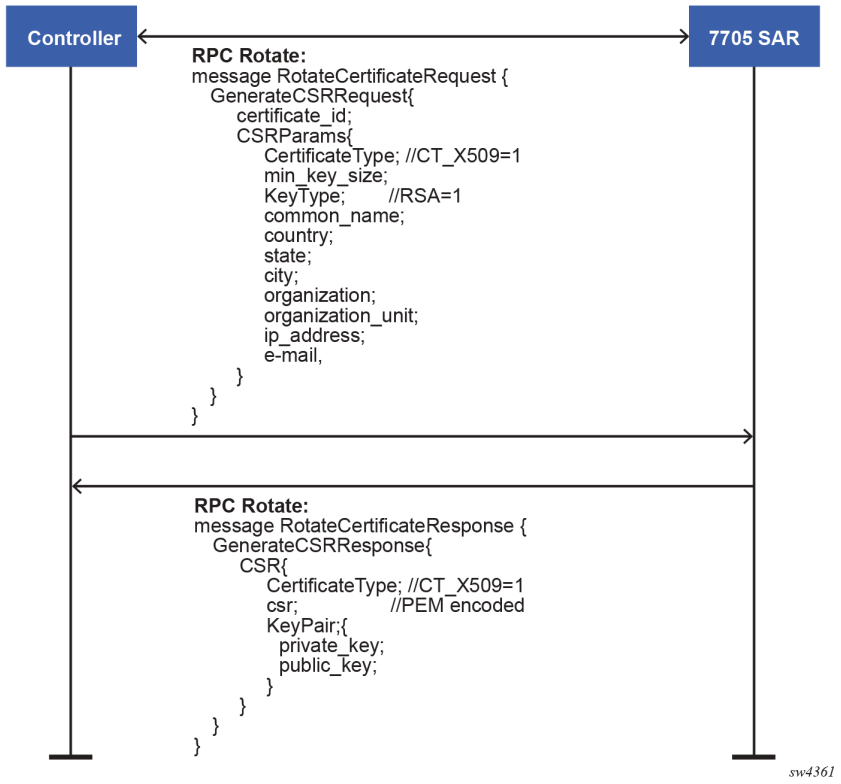


From the perspective of the interaction of the controller and the server (SR OS) two stages are the most important:

- message exchange to generate the CSR
- message exchange to load the new certificates on the server

The following figure shows a detailed content of the messages that are exchanged for CSR generation. The SR OS accepts requests only for the X509 certificate type, RSA key type, and a minimum key length of 512 bits.

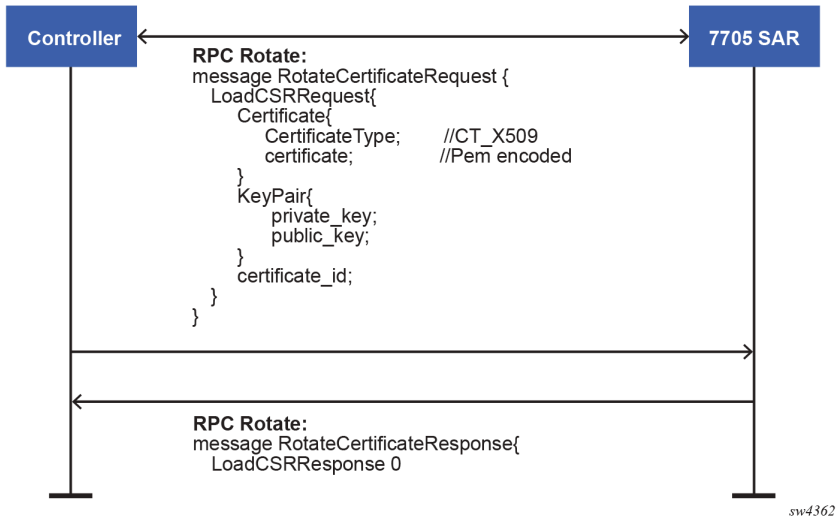
Figure 33: GenerateCSR message flow



For RPC Rotate, the certificate_id points to an existing certificate on the node. All the other parameters in the GenerateCSRRequest message are not checked by the SR OS software explicitly. They are used by the internal API to generate the CSR and that result is transparently passed to the controller.

After the CA signs the certificates, the files are loaded to the server using LoadCSRRequest and LoadCSRResponse message exchange, as shown in the following figure. If this message exchange is used in the context of RPC Rotate, the certificate_id should not be present in LoadCSRRequest message. When the SR OS receives the message, it performs all the necessary steps to load this certificate, including storing the certificate and key files on the disk.

Figure 34: LoadCSRRequest/Response message flow



The controller is responsible for verifying the connection with the new certificate (Step 4 in Figure 31: RPC Rotate message flow for CSRs generated on the SR OS node and Figure 32: RPC Rotate message flow when CSRs are not generated on the SR OS node); SR OS treats this as an optional step.

After the whole RPC is successfully closed, the system can use the new certificate to start new TLS connections.

10.3.1.4 RPC Install

The controller can use RPC Install to install a new certificate on the server. After the certificate is installed, the server must be configured (assign a certificate and key files in the CERT profile) before the new certificate can be used.

The following two possible cases are supported for installing a certificate:

- server capable of generating a CSR (see Figure 35: RPC install message flow for CSRs generated on the SR OS node)
- server is not capable of generating a CSR (see Figure 36: RPC install message flow if CSRs are not generated on the SR OS node)

The SR OS supports both scenarios, although it is assumed that in most cases the CSR is generated on the SR OS node.

Both scenarios require the following steps:

1. Generate the CSR.
2. Sign the CSR by the Certificate Authority (CA).
3. Load the new certificate on the server.

Figure 35: RPC install message flow for CSRs generated on the SR OS node

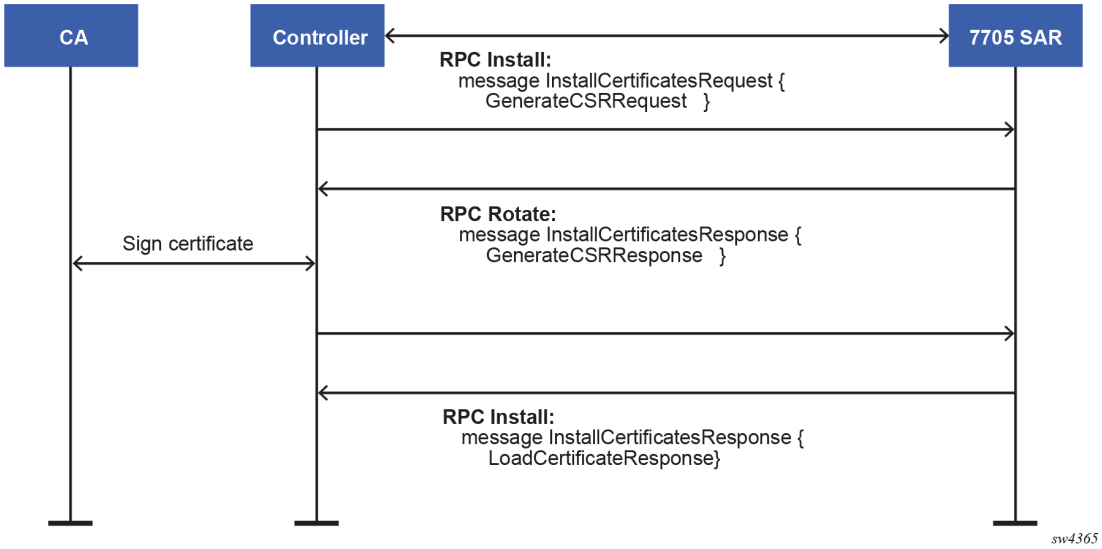
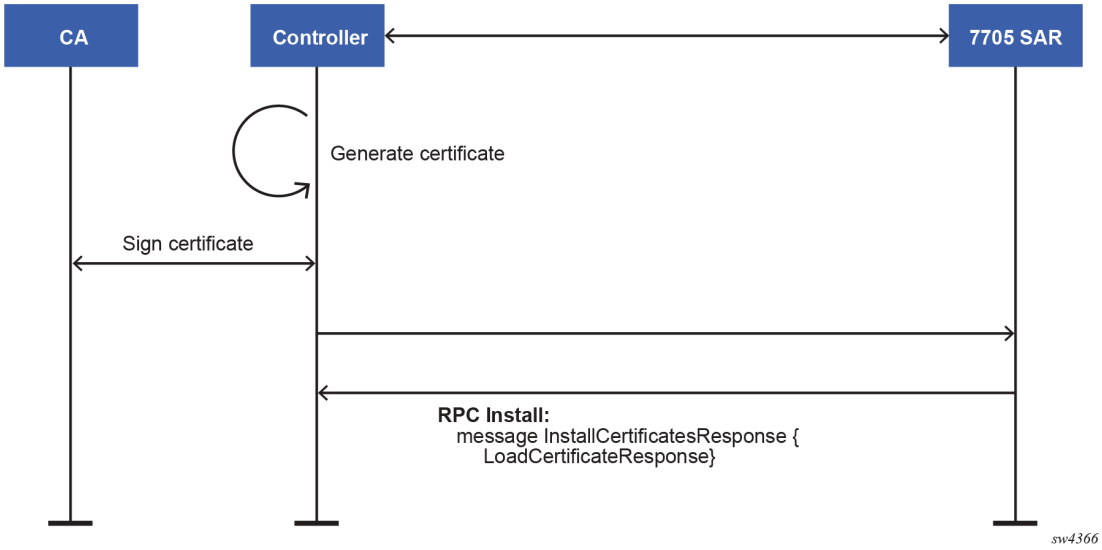


Figure 36: RPC install message flow if CSRs are not generated on the SR OS node



The message exchange during phases 1 and 3 is the same as shown in Figure 33: GenerateCSR message flow and Figure 34: LoadCSRRequest/Response message flow. The only difference, in the case of RPC Install, is that a new certificate_id is used.

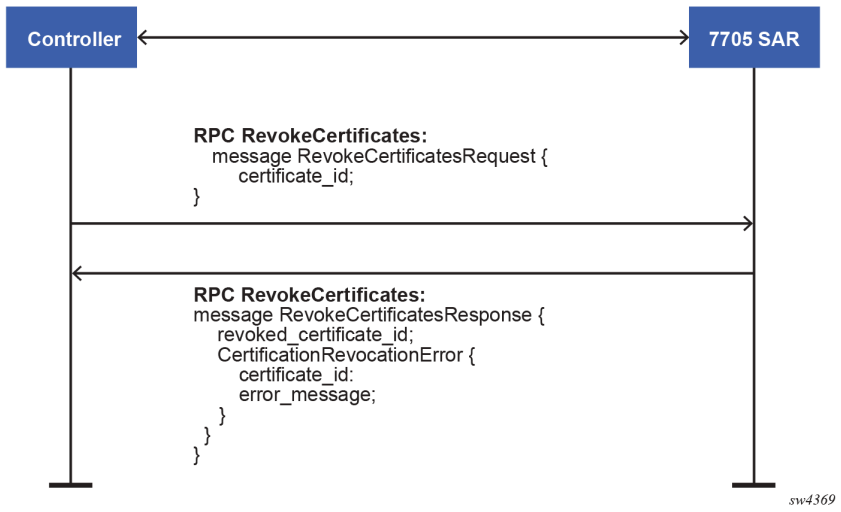
After new certificates are installed, the system must be configured before it can be used. Configuration is supported using the following methods:

- an existing gRPC session
- a CLI session, SNMP, or NETCONF

10.3.1.5 RPC RevokeCertificates

The purpose of the RPC RevokeCertificates is to make the existing certificate unusable by any client. In cases where the certificate being revoked by the client does not exist on the SR OS node, the corresponding RPC silently succeeds. The following figure shows the message flow.

Figure 37: RPC RevokeCertificates message flow



10.4 gNOI System

In the gNOI System service, OpenConfig defines a generic interface that performs operational tasks on target network nodes. The specification can be found at following link: *https://github.com/openconfig/gnoi/blob/master/system/system.proto*.

These operations can be performed on individual targets, regardless of vendor. SR OS supports the following gNOI system RPCs:

- SetPackage RPC
- Reboot RPC
- CancelReboot RPC
- RebootStatus RPC
- SwitchControlProcessor RPC
- Ping RPC
- Time RPC
- Traceroute RPC

10.4.1 SetPackage RPC

The SetPackage RPC allows the controller to place a software package on the target node. The file transfer is protected by the checksum. SR OS supports options where the controller can directly stream files to the target node. The remote download option is not supported.

The controller can use the SetPackage RPC to modify the `bof.cfg` file when the package destination is different from the bootable image path configured in `bof.cfg`.

10.4.2 Reboot, CancelReboot, and RebootStatus RPC

The Reboot RPC allows the controller to reboot a target node. The RebootRequest message can be used to specify reboot delay and system actions that should be performed during the reboot. SR OS supports only cold reboot; that is, the entire node is shut down and restarted during the reboot process. A Reboot RPC can be used to reboot both IOMs and CPMs.

The CancelReboot RPC allows the controller to cancel pending reboots.

The RebootStatus RPC allows the controller to query the status of a reboot for an individual component specified in the RebootStatus request. SR OS supports querying on a single component at a time.

10.4.3 SwitchControlProcessor RPC

The SwitchControlProcessor RPC allows the switching of the active Control-Processor to the Control-Processor that is provided in the request message. Because SR OS supports two Control-Processors, one of the following paths must be provided in the request message, depending on which Control-Processor is currently in standby.

1. `/state/cpm[cpm-slot=A]`
2. `/state/cpm[cpm-slot=B]`

10.4.4 Ping RPC

The Ping RPC allows the controller to execute the **ping** command on the target node and results are streamed back to the source node.

The user can specify a network instance to use while running the Ping RPC. If no network instance is specified, SR OS uses the Base network instance.

10.4.5 Time RPC

The Time RPC returns the current time on the target node. This RPC is typically used to test for a response from the target.

10.4.6 Traceroute RPC

The Traceroute RPC allows the controller to execute the **traceroute** command on the target node and results are streamed back to the source node.

The user can specify a network instance to use by running the Traceroute RPC. If no network instance is specified, SR OS uses the Base network instance.

10.5 gNOI file

In the gNOI file service, OpenConfig defines a generic interface to perform file operational tasks. For information about the gNOI specification, see the following link: https://github.com/openconfig/gnoi/blob/master/file/file.proto.

SR OS supports the following gNOI file RPCs:

- Get RPC
- Put RPC
- Stat RPC
- Remove RPC
- TransferToRemote RPC

Note: By default, the gNOI file RPCs are enabled in the user profile.

Use the commands in the following context to configure authorization for each file RPC:

- **MD-CLI**

```
configure system security aaa local-profiles profile grpc rpc-authorization
```

- **classic CLI**

```
configure system security profile grpc rpc-authorization
```

10.5.1 Get RPC

A Get RPC reads and streams the contents of a file from a target location. The file is streamed using sequential messages and a final message containing the hash of the streamed data is sent before the stream is closed. An error is returned when:

- the file does not exist
- there is a problem reading the file

10.5.2 Put RPC

A Put RPC streams data to be written on a file on the target location. The file is streamed using sequential messages and a final message that includes the hash of the streamed data is sent prior to closing the stream. An error is returned when:

- the location does not exist
- an error is encountered while writing the data

10.5.3 Stat RPC

A Stat RPC returns metadata (that is, statistical information) about a file on the target location. An error is returned when:

- the file does not exist
- an error is encountered while accessing the metadata

10.5.4 Remove RPC

A Remove RPC removes the specified file from the target location. An error is returned when:

- the file does not exist
- there is a directory instead of a file
- an error is encountered during the remove operation (for example, permission denied)

10.5.5 TransferToRemote RPC

A TransferToRemote RPC transfers the file from the target node to a specified remote location. When the file transfer is complete, the response contains the hash of the transferred data. An error is returned when:

- the file does not exist
- the file transfer fails
- an error is encountered reading the file

10.6 MD-CLI service

The SR OS provides a proprietary management interface to use with the Network Interface Shell (NISH) tool which allows an MD-CLI style interface from a remote location to manage one or more SR OS nodes.

This feature is applicable only on SR OS platforms that support MD-CLI in Model-Driven or mixed configuration mode.

This service operates using gRPC and, therefore, the main gRPC service must also be enabled.

When enabled, the MD-CLI gRPC service provides MD-CLI schema information to the NISH client allowing users to remotely operate the SR OS device.

The MD-CLI gRPC service and the main gRPC service must be enabled on all nodes that will be managed using the NISH client.

10.6.1 Remote management using a remote NISH manager

When used together with the MD-CLI gRPC service, the remote management feature allows SR OS nodes to initiate communication with a remote NISH manager and announce their availability to be managed using the NISH client. This provides the NISH client with a dynamic view of the available nodes that it can manage.

The remote management service does not perform or enable the actual management of the SR OS node using NISH. This communication is achieved directly from the NISH client to the SR OS node using the MD-CLI gRPC service.

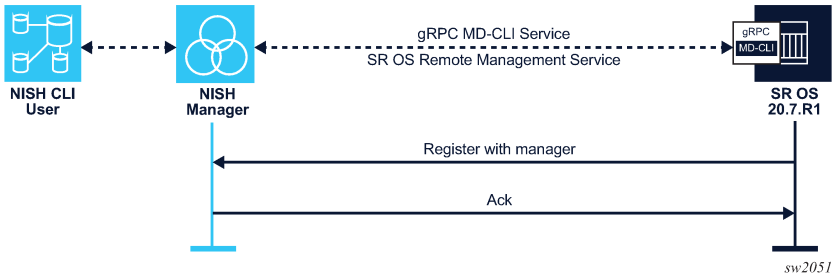
This feature is particularly useful when deploying clusters of SR OS nodes that may dynamically join or leave a cluster, such as in scenarios that use the Control and User Plane Separation (CUPS) BNG application with Virtualized Service Routers (VSRs).

A working NISH manager service is required on an external server to use the remote management feature. However, in the absence of a working NISH manager, the system will not stop remote management from being enabled within SR OS, nor will it stop the SR OS node from announcing its presence to the configured IP address or addresses of the NISH manager.

When a remote NISH manager is configured, the SR OS node initiates a gRPC session with the configured manager. The SR OS node sends a message to communicate its name, IP address (IPv4 and IPv6 are supported), and gRPC port to the NISH manager. The NISH manager responds with an acknowledgment message. The SR OS node periodically checks in with the NISH manager.

The following figure shows the remote management initiation.

Figure 38: Remote management service initiation



If the connection is interrupted, the SR OS node immediately attempts reconnection with the configured NISH manager.

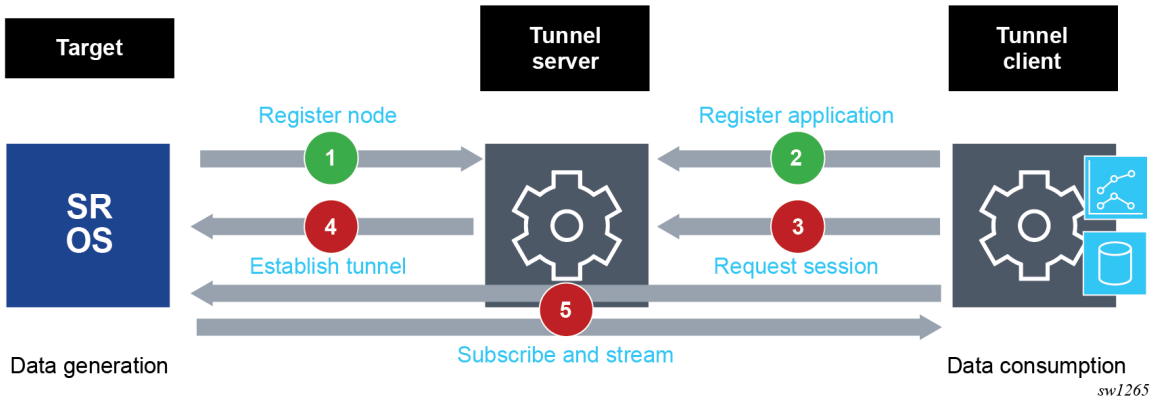
11 gRPC tunnels

A gRPC tunnel is a client-server protocol like any other gRPC-based service. The gRPC tunnel concept is defined by OpenConfig (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 all registered targets, along with information about supported target types, all registered tunnel clients and the target type they subscribe to.
- **tunnel clients**
  The tunnel client is a software entity which performs client tasks, such as requesting a session to the specified target.

The following figure shows the gRPC tunnel service concept.

Figure 39: gRPC tunnel service concept



After the registration process is complete, the tunnel server informs the tunnel client about all available targets supporting the target type that it is subscribed to. The tunnel client can then request a tunnel session toward a specific target. When a request is made, the tunnel server establishes a TCP tunnel between itself and the target (if one is not already open for another session) and establishes a tunnel session between the target and the tunnel client. After this session is established, the tunnel client can open any supported application session (gNMI or gNOI) toward applications on the target.

The following RPCs are defined to facilitate gRPC tunnels:

- **RegisterRPC**
  The RegisterRPC is used to perform the following tasks:
  - sent from a target to a tunnel server to register the network element after a reboot or the configuration of a gRPC tunnel on the network element
  - sent from a tunnel client to a tunnel server to subscribe to a specific target type
  - sent from a tunnel client to a tunnel server to request a session to a specified target with a specific target type

- **TunnelRPC**
  The TunnelRPC is used 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 RegisterRPC). The exchange of data on a TunnelRPC 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 from the gNOI client directly.

Using a gRPC tunnel instead of a direct gRPC connection provides the following advantages:

- Using a gRPC tunnel avoids firewall issues by initiating the TCP connection from the network element.
- The network element registers itself, which provides active network element discovery.
- The common gRPC tunnel interface on the network element does not require any adaptations to use different management interfaces, such as gNMI, gNOI, SSH, or NETCONF.

11.1 gRPC tunnels in SR OS

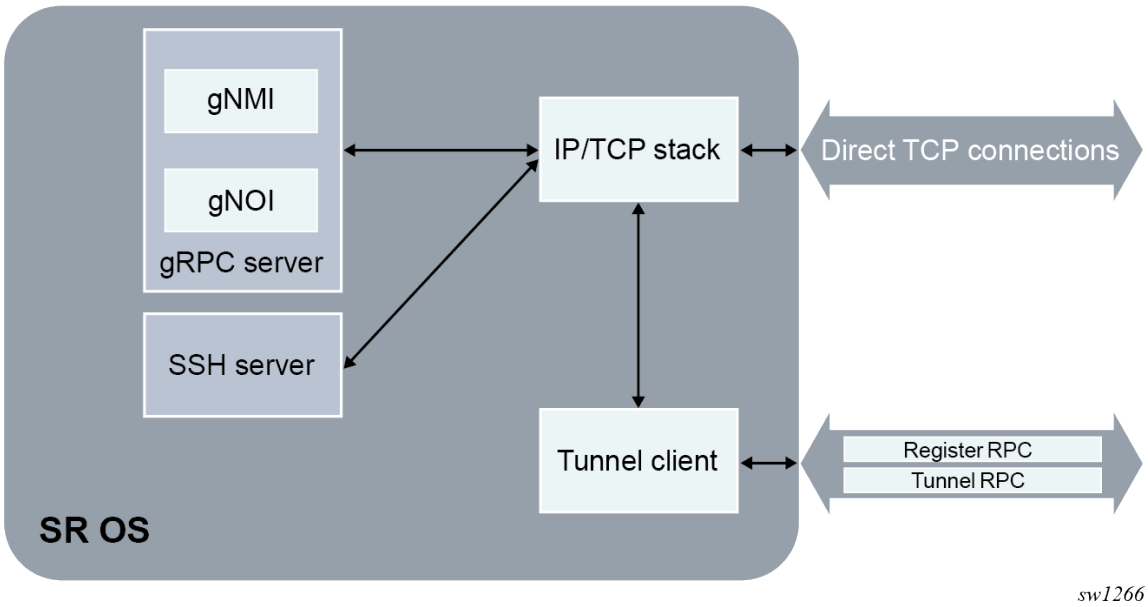
This section describes how to configure and use gRPC tunnels within SR OS. The SR OS gRPC implementation supports gRPC Tunnel version 0.2.

11.1.1 gRPC tunnel architecture in SR OS

The main function of a gRPC tunnel is to allow tunneling of TCP datagrams between SR OS nodes and third-party software entities. To minimize the impact on existing services, the internal cross-connect sends the decapsulated packets on a preconfigured TCP port using the local host as the destination. In this way, all internal applications receive the same data as they would from an external TCP connection.

The simplified view of the internal SR OS implementation is shown in the following figure.

Figure 40: Internal SR OS architecture



11.1.2 gRPC tunnel security

The OpenConfig gRPC tunnel specification requires TLS encryption at the tunnel and gRPC server level.

TLS encryption at tunnel level is configured by assigning the TLS client profile at the destination group level. It operates similarly to dial-out telemetry, as described in Dial-out telemetry.

When an external gRPC client connects to an SR OS 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.

11.1.3 Configuring a gRPC tunnel in SR OS

About this task

To configure use gRPC tunnel, perform the following steps:

Procedure

Step 1. Configure the destination group.

a. Configure 1 or 2 destinations using the **destination** command.

b. Configure whether transport is secure or unsecure using the **allow-secure-connection** command.

c. Configure the router instance using the **router-instance** command.

d. Optionally, configure a local source address using the **local-source-address** command.

e. Optionally, configure the TCP keepalive interval using the **tcp-keepalive** command.

Step 2. Configure the gRPC tunnel.

a. Assign a destination group using the **destination-group** command.

b. Enable the tunnel.

c. Define the target name using the **target-name** command.

d. Configure one or more handlers using the **handler** command.

Step 3. Configure the internal application server.

Example

MD-CLI

```
[ex:/configure system]
A:admin@node-2# info
    grpc-tunnel {
        destination-group "ba-server" {
            allow-unsecure-connection
            destination 1.1.1.1 port 33333 {
                router-instance "management"
            }
        }
        tunnel “test” {
            admin-state enable
            destination-group “ba-server”
            target-name { node-name}
```

```
            handler “my-grpc” {
                admin-state enable
                target-type { grpc-server }
                port 57400
            }
            handler “ssh” {
                admin-state enable
                target-type { ssh-server }
                port 22
            }
        }
    }
    grpc {
        admin-state enable
        allow-unsecure-connection
        gnmi {
            admin-state enable
        }
    }
```

Example

classic CLI

```
A:node-2>config>system# info
     grpc-tunnel
            destination-group "ba-server" create
                allow-unsecure-connection
                tcp-keepalive
                    shutdown
                exit
                destination 1.1.1.1 port 33333 create
                    router-instance "management"
                exit
            exit
            tunnel "test" create
                destination-group "ba-server"
                target-name node-name
                handler "ssh" create
                    port 22
                    target-type ssh-server
                    no shutdown
                exit
                handler "my-grpc" create
                    port 57400
                    target-type grpc-server
                    no shutdown
                exit
                no shutdown
            exit
        exit

     grpc
            allow-unsecure-connection
            no shutdown
            gnmi
                no shutdown
            exit
      exit
```

11.1.4 Verifying gRPC tunnel operation

About this task

This procedure describes the commands used to verify gRPC tunnel operation.

Procedure

Step 1. Use the following command to view the state of a GRPC tunnel.

```
show system grpc-tunnel tunnel tunnel-name
```

Example

```
===============================================================
gRPC-Tunnel tunnel
===============================================================================
Name                      : test
Administrative State      : Enabled
Operational State         : Up
Description               : (Not Specified)
Destination Group         : ba-server
Operational Target Name   : DUT-A
===============================================================================
```

Step 2. Use the following command to view destinations for a GRPC tunnel.

```
show system grpc-tunnel tunnel tunnel-name destinations
```

Example

```
===============================================================
gRPC-Tunnel tunnel
===============================================================================
Name                      : test
Administrative State      : Enabled
Operational State         : Up
Description               : (Not Specified)
Destination Group         : ba-server
Operational Target Name   : DUT-A

-------------------------------------------------------------------------------
Destinations
-------------------------------------------------------------------------------
Destination               : 1.1.1.1
Port                      : 33333
Operational State         : Up
Last Oper Down Reason     : MINOR: GMI #2452: RPC refused by peer - error code
                            14 (UNAVAILABLE), error message: Connect Failed
Oper Router Instance      : management
Last Oper Change          : 2022/01/13 14:59:06
Connection Attempts       : 11469
-------------------------------------------------------------------------------
No. of destinations       : 1
===============================================================================
```

12 TLS

Transport Layer Security (TLS) is used for the following primary purposes:

- **authentication of an end device (client or server) using a digital signature (DS)**

  TLS uses Public Key Infrastructure (PKI) for device authentication. DSs are used to authenticate the client or the server. The server typically sends a certificate with a DS to the client.

  In certain situations, the server can request a certificate from the client to authenticate it. The client has a certificate (called a trust anchor) from the certificate authority (CA), which is used to authenticate a server certificate and its DS. After the client provides a digitally signed certificate to the server and both parties are authenticated, the encryption Protocol Data Units (PDUs) are transmitted.

  When the SR OS, acting as a server, requests a certificate from the client, the client must provide the certificate. If the client fails to provide a certificate for authentication, the SR OS terminates the TLS session. The server TLS settings can be configured to not request certificates, in which case the client is not required to send the server a certificate for authentication.

- **encryption and authentication of application PDUs**

  After the clients and server are successfully authenticated, the cipher suite is negotiated between the server and clients, and the PDUs are encrypted based on the agreed cipher protocol.

12.1 TLS server interaction with applications

TLS is a standalone configuration. The user must configure TLS server profiles with certificates and trust anchors, and then assign the TLS server profiles to the appropriate applications. When a TLS server profile is assigned to an application, the application should not send any clear text PDUs until the TLS handshake is successfully completed and the encryption ciphers are negotiated between the TLS server and the TLS client.

After successful negotiation and handshake, the TLS is operationally up and notifies the application, which begins transmitting PDUs. These PDUs are encrypted using TLS based on the agreed ciphers. At any point, if the TLS becomes operationally down, the application stops transmitting PDUs.

For example, the following sequence describes a TLS connection with the gRPC application:

1. A TLS server profile is assigned to the gRPC application.
2. The gRPC stops sending clear text PDUs because a TLS server profile has been assigned and TLS is not ready to encrypt.
3. The TLS server begins the handshake.
4. Authentication occurs at the TLS layer.
5. The TLS server and TLS client negotiate ciphers.
6. Salts are negotiated for the symmetric key. A salt is a seed for creating Advanced Encryptions Standard (AES) encryption keys.
7. When negotiations are successfully completed, the handshake finishes and the gRPC is notified.

8. TLS becomes operationally up, and the gRPC can resume transmitting PDUs. Until TLS is operationally up, gRPC PDUs arriving from the client are dropped on ingress.

12.1.1 TLS application support

The following table lists the applications that support TLS.

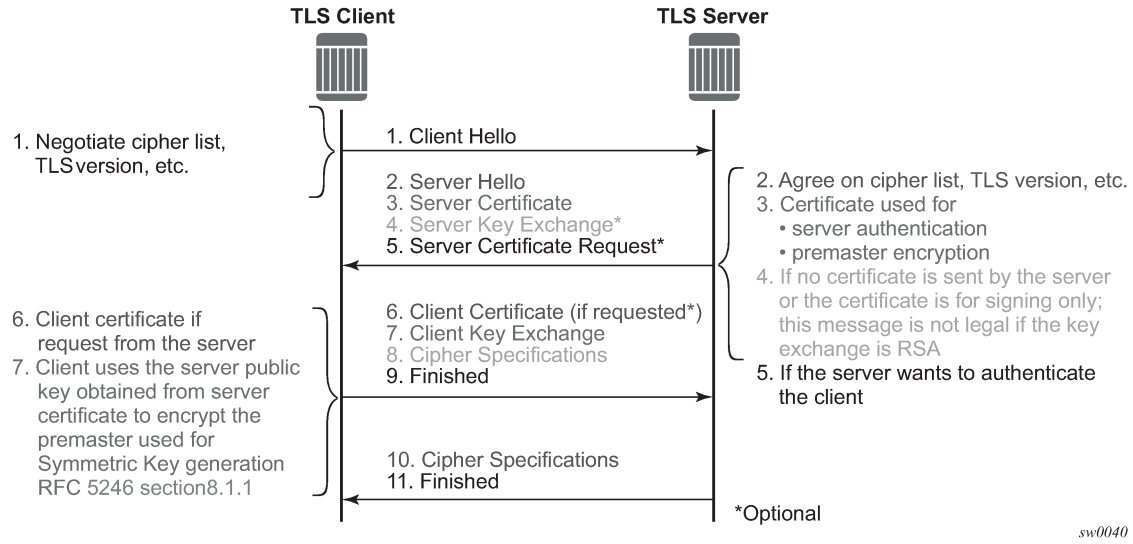
Table 56: TLS application support

Application	TLS server supported	TLS client supported
gRPC	✓	✓
LDAP		✓
RADIUS		✓
Syslog		✓

12.2 TLS handshake

The following figure shows the TLS handshake.

Figure 41: TLS handshake



The following table describes the steps in the TLS handshake.

Table 57: TLS handshake step descriptions

Step	Description
1	The TLS handshake begins with the client Hello message. This message includes the cipher list that the client wants to use and negotiate, among other information.
2	The TLS server sends back a server Hello message, along with the first common cipher found on both the client and server cipher lists. The common cipher is used for data encryption.
3	The TLS server sends a server certificate message, in which the server provides a certificate that the client can use to authenticate the server identity. The public key of this certificate (RSA key) can also be used to encrypt the symmetric key seed used by the client and server to create the symmetric encryption key. This occurs only if the PKI is using RSA for asymmetric encryption.
4	SR OS does not support server key exchange. SR OS uses only RSA keys; Diffie-Hellman key exchange is not supported.
5	The server can optionally be configured to request a certificate from the client to authenticate the client.
6	If the server requests a client certificate, the client must provide it by using a client certificate message. If the client does not respond to the certificate request, the server drops the TLS session.
7	The client uses the public server RSA key included in the server certificate to encrypt a seed. The client and server use this seed to create the identical symmetric key used for encrypting and decrypting data plane traffic.
8	The client sends a cipher spec message to switch encryption to this symmetric key.
9	The client successfully finishes the handshake.
10	The server sends a cipher spec message to switch encryption to this symmetric key.
11	The server successfully finishes the handshake.

After a successful handshake, TLS is operationally up and applications can use TLS for application encryption.

12.3 TLS 1.3

TLS 1.3 is required for faster handshakes and stronger encryption and authentication algorithms.

All SR OS applications that use TLS 1.2 also support TLS 1.3, unless specifically stated otherwise.

The user can configure the node to use TLS 1.2, TLS 1.3, or both for its client or server negotiation.

When TLS 1.3 is negotiated with a client, the node no longer negotiates the TLS version down to 1.2 as long as the session is alive.

12.3.1 TLS 1.3 handshake

The TLS 1.3 client handshake is very similar to TLS 1.2 because the client is able to negotiate TLS 1.2 or 1.3 when starting the TLS Hello message to the server. The client includes a "Supported Version" extension in its Hello message. The server responds with its own supported version, agreed ciphers, and so on.

In TLS 1.2 and TLS 1.3, the server can optionally request for the client certificate to authenticate the client. If requested, the client must provide its certificate to the server.

12.3.2 TLS 1.3 configuration

The user can configure the TLS 1.3 cipher list independently of TLS 1.2 for both client and server ciphers. TLS 1.3 ciphers are configured using the **tls13-cipher** command.

TLS 1.3 also introduces group lists and signature lists for the server and client.

In the Hello message sent by the client, the "supported_groups" extension indicates the named groups that the client supports for the key exchange, ordered from most preferred to least preferred. TLS 1.3 supports Elliptic-Curve Diffie-Hellman Ephemeral (ECDHE) groups.

Note: TLS 1.2 does not support Diffie-Hellman groups as an asymmetric key.

TLS 1.3 also allows the selection of signature algorithms. The "signature_algorithms_cert" extension is included to allow implementations that support different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities.

When the user configures a TLS 1.3 cipher list, the TLS handshake includes TLS 1.3 as a supported TLS version in the TLS handshake.

12.4 TLS client certificate

The TLS protocol is used for authentication. It allows the server to authenticate the client via PKI. If the server requests authentication from the client, the client response must provide an X.509v3 certificate that the server can authenticate using the digital signature of its client. The SR OS supports the configuration of an X.509v3 certificate for TLS clients. When the server requests a certificate using the Hello message, the client sends a client certificate message to transmit its certificate to the server.

12.5 Certificate revocation status verification for TLS

A certificate authority (CA) can revoke issued certificates by listing them in a certificate revocation list (CRL). In TLS, an optional CRL is applied for CA certificates. The CRL does not apply to the intermediate

or end issuer of an end entity (EE) certificate. To extend this optional configuration to include EE certificates, use the **status-verify default-result** commands under the following contexts:

```
configure system security tls client-tls-profile
configure system security tls server-tls-profile
```

The command options are **revoked** (default value) or **good**.

This default result is used when the revocation status of a certificate cannot be determined because of an invalid CRL (for example, it is missing, expired, or corrupt).

The TLS **default-result** for EE certificates is set to **revoked** for safety purposes. If an expired CRL in the CA profile is matched as the issuer of the EE certificate, the EE certificate is treated as revoked. If the expired CRL is further up in the certificate chain, the optional CRL works as expected.

The **status-verify default-result** commands allow users to override the recommended revocation check policy when there is legitimate reason to accept EE certificates without checking their revocation status (for example, to keep the automatic CRL update working during a temporary network issue).

12.6 EE revocation status verification using OCSP

The OCSP method to determine the revocation status of TLS certificates is currently supported only for the gRPC application.

Note: OCSP is not supported for any other application over TLS (for example, LDAP, Syslog, PCEP, OpenFlow). OCSP is supported only for end-entity (EE) certificates; it is not supported for CA certificates.

The default primary revocation option is **crl** and the default secondary revocation option is **none**. Use the following command to configure an alternative supported option, such as **ocsp**, for the EE revocation method.

```
configure system security tls client-tls-profile status-verify ee-revocation primary ocsp
 secondary ocsp
```

Note: The CA revocation is supported only via CRL.

To configure the OCSP responder, use the commands in the following context.

```
configure system security pki ca-profile ocsp
```

The responses from the OCSP responder (also known as the OCSP server) are cached in SR OS. After the gRPC connection is opened, all gRPC TLS handshakes are verified using the cached information. Verification is not performed via the OCSP server until the cached information is removed.

The cached information is removed periodically. The gRPC TLS handshakes that occur after the removal are verified again via the OCSP server. SR OS checks the cached TLS certificate information and if the validity information of the TLS certificate is not in the cache, sends a request to the OCSP server.

To validate the EE certificate, OCSP sends a blocking call to the OCSP server.

Note: While the EE certificate validation is in progress, OCSP blocks the application. The OCSP timeout should be configured to ensure that the application is not affected. The gRPC task is

blocked during the OCSP verification of the TLS certificate in the TLS handshake. Therefore, Nokia recommends that the configured timeout be as small as possible.

Use the following command to configure the timeout for the OCSP server response:

- **MD-CLI**

```
configure system transmission-profile timeout
```

- **classic CLI**

```
configure system file-transmission-profile timeout
```

12.7 TLS symmetric key rollover

The SR OS supports key rollover using HelloRequest messages, in accordance with RFC 5246, section 7.4.1.1. Some applications have a longer live time than other applications, in which case the SR OS can use a timer that prompts the HelloRequest negotiation for symmetric key rollover. This timer can be configured using CLI.

Use the following command to configure the TLS re-negotiate timer.

```
configure system security tls server-tls-profile tls-re-negotiate-timer
```

You can disable the timer configuration for applications that do not support HelloRequest messages (for example, the gRPC application).

When the **tls-re-negotiate-timer** command is configured, the HelloRequest message is not generated and symmetric keys are not renegotiated.

12.8 Supported TLS ciphers

As shown in Figure 41: TLS handshake, TLS negotiates the supported ciphers between the client and the server.

The client sends the supported cipher suites in the client Hello message, and the server compares them with the server cipher list. The top protocol on both lists is chosen and returned from the server within the server Hello message.

SR OS supports the following TLS 1.2 ciphers as a TLS client or TLS server:

- tls-rsa-with3des-ede-cbc-sha
- tls-rsa-with-aes128-cbc-sha
- tls-rsa-with-aes256-cbc-sha
- tls-rsa-with-aes128-cbc-sha256
- tls-rsa-with-aes256-cbc-sha256
- tls-rsa-with-aes128-gcm-sha256
- tls-rsa-with-aes256-gcm-sha384
- tls-ecdhe-rsa-aes128-gcm-sha256

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

SR OS supports the following TLS 1.3 ciphers, groups, and signature algorithms as a TLS client or TLS server:

- tls-aes128-gcm-sha256
- tls-aes256-gcm-sha384
- tls-chacha20-poly1305-sha256
- tls-aes128-ccm-sha256
- tls-aes128-ccm8-sha256
- **Groups:**
  - tls-ecdhe-256
  - tls-ecdhe-384
  - tls-ecdhe-521
  - tls-x25519
  - tls-x448
- **Signature algorithms:**
  - tls-rsa-pkcs1-sha256
  - tls-rsa-pkcs1-sha384
  - tls-rsa-pkcs1-sha512
  - tls-ecdsa-secp256r1-sha256
  - tls-ecdsa-secp384r1-sha384
  - tls-ecdsa-secp521r1-sha512
  - tls-rsa-pss-rsae-sha256
  - tls-rsa-pss-rsae-sha384
  - tls-rsa-pss-rsae-sha512
  - tls-rsa-pss-pss-sha256
  - tls-rsa-pss-pss-sha384
  - tls-rsa-pss-pss-sha512
  - tls-ed25519
  - tls-ed448

12.9 SR OS certificate management

SR OS implements a centralized certificate management protocol that can be used by TLS and IPsec.

Use the commands in the following contexts to configure and manage certificates:

- **MD-CLI**

```
admin system security pki
configure system security pki
```

- **classic CLI**

```
admin certificate
configure system security pki
```

12.9.1 Certificate profile

The certificate profile is available for both the TLS server and the TLS client. The **cert-profile** command is configured for the server or client to transmit the provider certificate and its DS to the peer so that the peer can authenticate it via the **trust-anchor** and CA certificate.

Multiple provider certificates can be configured on the SR OS; however, the SR OS currently uses the smallest index as the active provider certificate, and only sends the certificate to the peer.

12.9.2 TLS server authentication of the client certificate CN field

If the client provides a certificate upon request by the server, the SR OS checks the certificate common name (CN) field against local CN configurations. The CN is validated via the client IPv4 or IPv6 address, or FQDN.

If the CN list is not configured using the following command, the SR OS does not authenticate via the CN field and relies only on certificate signature authentication:

- **MD-CLI**

```
configure system security tls server-tls-profile authenticate-client common-name-list
```

- **classic CLI**

```
configure system security tls server-tls-profile authenticate-client cn-authentication
```

12.9.3 CN regexp format

Use commands in the following context to configure CN entries.

```
configure system security pki common-name-list
```

Entries should use regular expression (regexp), FQDN, or the IP address.

For information about regexp, see the *7705 SAR Gen 2 Classic CLI Command Reference Guide*, Entering CLI commands section.

12.10 Operational guidelines

This section provides operational guidelines for TLS.

12.10.1 Server authentication behavior

Following the Hello messages, if the certificate must be authenticated, the server sends its certificate in a certificate message. If required, a ServerKeyExchange message may also be sent. See RFC 5246, section 7.3 for more information about the authentication behavior on the LDAP server.

Use the commands in the following context to configure client TLS security. The **trust-anchor-profile** command determines whether the server must be authenticated by the client.

```
configure system security tls client-tls-profile trust-anchor-profile
```

Note: If the **trust-anchor-profile** command is configured and the **ca-certificate** or **ca-profile** is missing from this **trust-anchor-profile**, the TLS connection fails and an "unknown_ca" error is generated, as defined in RFC 5246 section 7.2.2.

One of the following configurations can be used to establish server connectivity:

- If the **trust-anchor-profile** command is configured under the **configure system security tls client-tls-profile** context, the server must be authenticated using the **trust-anchor-profile** command before a trusted connection is established between the server and the client.
- If there is no **trust-anchor-profile** command under the **configure system security tls client-tls-profile** context, the trusted connection can be established without server authentication. The RSA key of the certificate is used for public key encryption, requiring the following basic checks to validate the certificate:
  - **time validity**

    The certificate is checked to ensure that it is neither expired nor not yet valid.
  - **certificate type**

    The certificate is not a CA certificate.
  - **keyUsage extension**

    If present, this must contain a digital signature and key encryption.
  - **host verification**

    The IP address or DNS name of the server is looked up, if available (for LDAP, only the IP address is used), in the common name (cn) or subjectAltName extension. This is to verify that the certificate was issued to that server and not to another.

12.10.2 Client TLS profile and trust anchor behavior and scale

The SR OS supports the creation of client TLS profiles, which can be assigned to applications such as LDAP to encrypt the application layer.

The **client-tls-profile** command is used for negotiating and authenticating the server. After the server is authenticated via the trust anchor profile (configured using the **trust-anchor-profile** command) of a client TLS profile, the server negotiates the ciphers and authentication algorithms to use for data encryption.

The client TLS profile must be assigned to an application for it to start encrypting. Up to 16 client TLS profiles can be configured. Because each of these client TLS profiles needs a trust anchor profile to authenticate the server, up to 16 trust anchor profiles can be configured. A trust anchor profile holds up to 8 trust anchors (configured using the **trust-anchor** command), each of which holds a CA profile (**ca-profile**).

A CA profile is a container for installing CA certificates (**ca-certificates**). The CA certificates are used to authenticate the server certificate. When the client receives the server certificate, the client reads through the trust anchor profile CA certificates and tries to authenticate the server certificate against each CA certificate. The first CA certificate that authenticates the server is used.

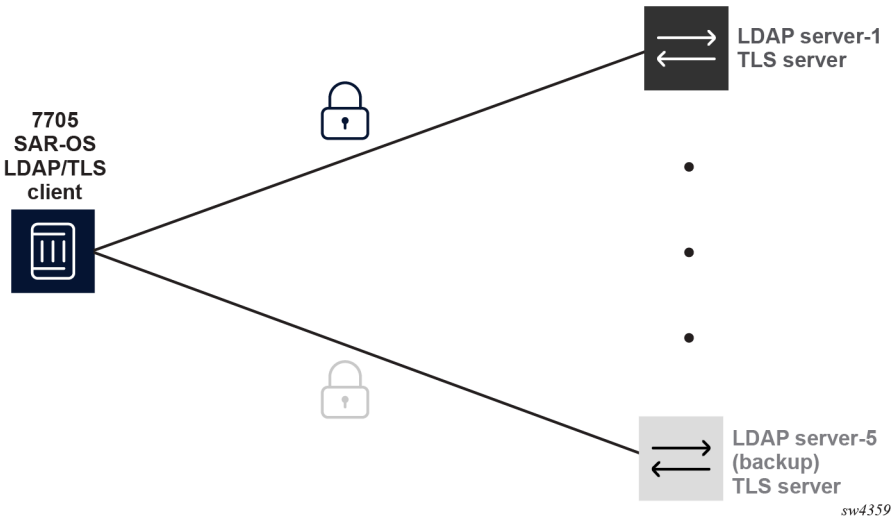
12.11 LDAP redundancy and TLS

LDAP supports up to five redundant (backup) servers. Depending on the configuration of **timeout** and **retry** values, if an LDAP server is found to be out of service or operationally down, the SR OS will switch to the redundant servers. The SR OS will try the next LDAP server in the serred server index.

LDAP servers can use the same TLS profile or can have their own TLS profile. Each TLS profile can have a different configuration of **trust-anchor** and **cert-profile**. For security reasons, the LDAP server could be in different geographical areas and, therefore, each will be assigned its own server certificate and trust anchor. The TLS profile design allows users to mix and match all components.

Redundant LDAP servers are shown in Figure 42: LDAP and TLS redundancy.

Figure 42: LDAP and TLS redundancy



12.12 Basic TLS configuration

Basic TLS server configuration requires the following:

- Use the following command to create a cipher list.

```
configure system security tls server-cipher-list
```

- Use the following command to assign the cipher list to the TLS server profile.

```
configure system security tls server-tls-profile cipher-list
```

- Use the following command to create a certificate profile.

```
configure system security tls cert-profile
```

- Use the following command to assign the certificate profile to the TLS server profile.

```
 configure system security tls server-tls-profile cert-profile
```

Basic TLS client configuration requires the following:

- Use the following command to create a cipher list.

```
configure system security tls client-cipher-list
```

- Use the following command to assign a TLS client to the TLS client profile.

```
configure system security tls client-tls-profile cipher-list
```

TLS imports the trust anchor certificate for (TLS) peer certificate authentication and public key retrieval.

The following example shows a TLS configuration.

Example: MD-CLI

```
[ex:/configure system security tls]
A:admin@node-2# info
    client-cipher-list "to-active-server" {
        tls12-cipher 1 {
            name tls-rsa-with-aes256-cbc-sha256
        }
        tls12-cipher 2 {
            name tls-rsa-with-aes128-cbc-sha256
        }
        tls12-cipher 3 {
            name tls-rsa-with-aes256-cbc-sha
        }
    }
    client-tls-profile "server-1-profile" {
        admin-state enable
        cipher-list "to-active-server"
        trust-anchor-profile "server-1-ca"
    }
    trust-anchor-profile "server-1-ca" {
        trust-anchor "tls-server-1-ca" { }
    }
```

Example: classic CLI

```
A:node-2>config>system>security>tls# info
----------------------------------------------
        trust-anchor-profile "server-1-ca" create
```

```
            trust-anchor "tls-server-1-ca"
        exit
        client-cipher-list "to-active-server" create
            cipher 1 name tls-rsa-with-aes256-cbc-sha256
            cipher 2 name tls-rsa-with-aes128-cbc-sha256
            cipher 3 name tls-rsa-with-aes256-cbc-sha
        exit
        client-tls-profile "server-1-profile" create
            cipher-list "to-active-server"
            trust-anchor-profile "server-1-ca"
            no shutdown
        exit
-------------------------------------------
```

12.13 Common configuration tasks

This section provides information about common configuration tasks.

12.13.1 Configuring a server TLS profile

Use the commands in the following context to configure a TLS server profile.

```
configure system security tls server-tls-profile
```

12.13.2 Configuring a client TLS profile

Use the following command to configure a client TLS profile, which also configures the server authentication behavior.

```
configure system security tls client-tls-profile
```

12.13.3 Configuring a TLS client or TLS server certificate

Use the following commands to configure TLS certificate management.

```
configure system security tls cert-profile
configure system security tls client-tls-profile cert-profile
configure system security tls server-tls-profile cert-profile
```

12.13.4 Configuring a TLS trust anchor

Use the commands in the following contexts to configure a TLS trust anchor.

```
configure system security pki ca-profile
configure system security pki certificate-display-format
configure system security tls trust-anchor-profile
configure system security tls client-tls-profile
```

The following example shows a TLS trust anchor configuration.

Example: MD-CLI

```
[ex:/configure system security pki]
A:admin@node-2# info
    ca-profile "tls-server-1-ca" {
        admin-state enable
        cert-file "tls-1-Root-CERT"
        crl-file "tls-1-CRL-CERT"
    }

[ex:/configure system security tls]
A:admin@node-2# info
    client-tls-profile "server-1-profile" {
        admin-state enable
        cipher-list "to-active-server"
        trust-anchor-profile "server-1-ca"
    }
    trust-anchor-profile "server-1-ca" {
        trust-anchor "tls-server-1-ca" { }
    }
```

Example: classic CLI

```
A:node-2>config>system>security>pki# info
----------------------------------------------
        ca-profile "tls-server-1-ca" create
            cert-file "tls-1-Root-CERT"
            crl-file "tls-1-CRL-CERT"
            no shutdown
        exit
----------------------------------------------
A:node-2>config>system>security>tls# info
----------------------------------------------
        trust-anchor-profile "server-1-ca" create
            trust-anchor "tls-server-1-ca"
        exit
        client-tls-profile "server-1-profile" create
            cipher-list "to-active-server"
            trust-anchor-profile "server-1-ca"
            no shutdown
        exit
```

13 Facility alarms

This chapter provides information about facility alarms.

13.1 Facility alarms overview

Facility alarms provide a tool for operators to track and display the basic status of their equipment facilities.

Facility Alarm support is intended to cover a focused subset of router states that are likely to indicate service impacts (or imminent service impacts) related to the overall state of hardware assemblies (cards, fans, links, and so on).

In the CLI, for brevity, the keyword or command **alarms** is used for commands related to facility alarms. This chapter may occasionally use the term alarm or alarms as a short form for facility alarms.

CLI display (**show** routines) allows the system operator to identify current facility alarm conditions and recently cleared alarms without searching event logs or monitoring various card and port show commands to determine the health of managed objects in the system, such as cards and ports.

The SR OS alarm model is based on RFC 3877, *Alarm Management Information Base (MIB)*, (which evolved from the IETF DISMAN drafts).

13.2 Facility alarms versus log events

Facility Alarms are different from log events. Facility alarms have a state (at least two states: active and clear) and a duration, and can be modeled with state transition events (raised, cleared). A log event occurs when the state of some object in the system changes. Log events notify the operator of a state change (for example, a port going down, an IGP peering session coming up, and so on). Facility alarms show the list of hardware objects that are currently in a bad state. Facility alarms can be examined at any time by an operator, whereas log events can be sent by a router asynchronously when they occur (for example, as an SNMP notification or trap, or a syslog event).

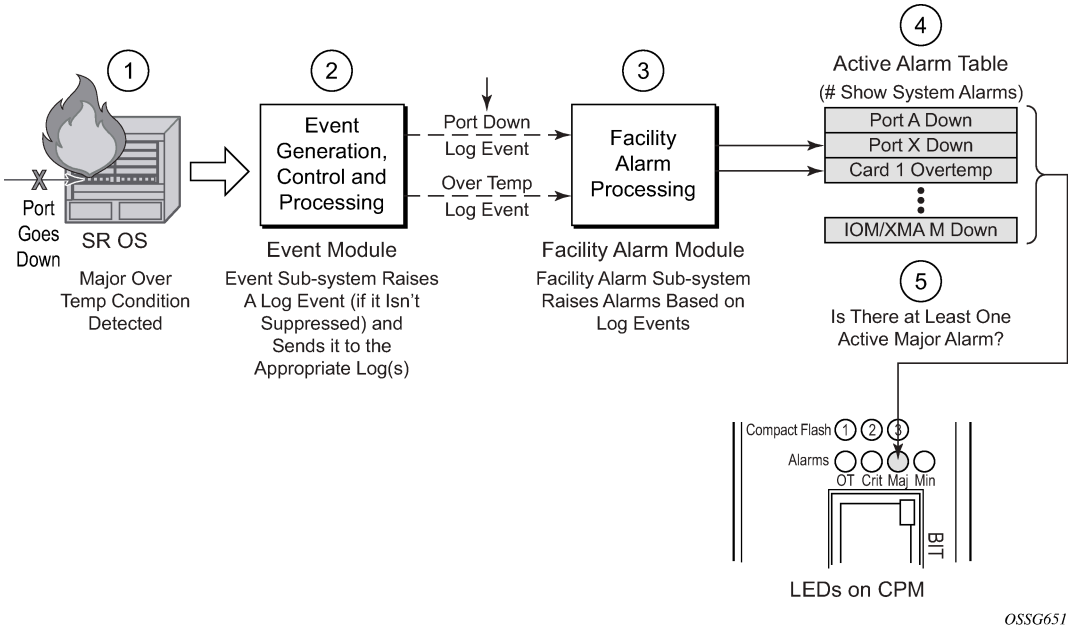
While log events provide notifications about a large number of different types of state changes in SR OS, facility alarms are intended to cover a focused subset of router states that are likely to indicate service impacts (or imminent service impacts) related to the overall state of hardware assemblies (cards, fans, links, and so on).

The facility alarm module processes log events to generate the raised and cleared state for the facility alarms. If a raising log event is suppressed under event-control, then the associated facility alarm is not raised. If a clearing log event is suppressed under event-control, then it is still processed for the purpose of clearing the associated facility alarm. If a log event is a raising event for a Facility Alarm, and the associated Facility Alarm is raised, then changing the log event to **suppress** clears the associated Facility Alarm.

Log event filtering, throttling and discarding of log events during overload do not affect facility alarm processing. In all cases, non-suppressed log events are processed by the facility alarm module before they are discarded.

Figure 43: Log events, facility alarms and LEDs illustrates the relationship of log events, facility alarms and the LEDs.

Figure 43: Log events, facility alarms and LEDs



Facility alarms are different and have independent functionality from other uses of the term alarm in SR OS such as:

- log events that use the term alarm (tmnxEqPortSonetAlarm)
- alarm configuration in the following contexts

```
configure card fp hi-bw-mcast-src alarm
configure multicast-management multicast-info-policy bundle channel source-override video
 analyzer alarms
configure port ethernet report-alarm
configure system thresholds rmon alarm
configure system security cpu-protection policy alarm
```

- memory-use alarms:
  - **MD-CLI**

    ```
    configure system thresholds kb-memory-use-alarm
    ```

  - **classic CLI**

    ```
    configure system thresholds memory-use-alarm
    ```

13.3 Facility alarm severities and alarm LED behavior

The alarm LEDs on the CPM/CCM reflects the current status of the facility alarms:

- The critical alarm LED is lit if there is 1 or more active critical facility alarms
- Similarly with the Major and Minor alarm LEDs
- The OT alarm LED is not controlled by the facility alarm module

The supported alarm severities are as follows:

- Critical (with an associated LED on the CPM/CCM)
- Major (with an associated LED on the CPM/CCM)
- Minor (with an associated LED on the CPM/CCM)
- Warning (no LED)

Facility alarms inherit their severity from the raising log event.

A raising log event for a facility alarm configured with a severity of *indeterminate* or *cleared* results in the facility alarm not being raised. But, a clearing log event is processed to clear facility alarms, regardless of the severity of the clearing log event.

Changing the severity of a raising log event only affects subsequent occurrences of that log event and facility alarms. Facility alarms that are already raised when their raising log event severity is changed maintain their original severity.

13.4 Facility alarm hierarchy

Facility alarms for children objects is not raised for failure of a parent object. For example, when an MDA or XMA fails (or is shutdown) there is not a set of port facility alarms raised.

When a parent facility alarm is cleared, children facility alarms that are still in occurrence on the node appears in the active facility alarms list. For example, when a port fails there is a port facility alarm, but if the MDA or XMA is later shutdown the port alarm is cleared (and a card alarm is active for the MDA or XMA). If the MDA or XMA comes back into service, and the port is still down, then a port alarm becomes active again.

The supported facility alarm hierarchy is as follows (parent objects that are down cause alarms in all children to be masked):

- CPM -> Compact Flash
- CCM -> Compact Flash
- IOM/IMM -> MDA -> Port -> Channel
- XCM -> XMA -> Port

Note: A masked facility alarm is not the same as a cleared facility alarm. The cleared facility alarm queue does not display entries for previously raised facility alarms that are currently masked. If the masking event goes away, then the previously raised facility alarms are visible again in the active facility alarm queue.

13.5 Facility alarm list

Table 58: Facility alarm, facility alarm name, raising log event, sample details string and clearing log event and Table 59: Facility alarm name/raising log event, cause, effect and recovery show the supported facility alarms.

Table 58: Facility alarm, facility alarm name, raising log event, sample details string and clearing log event

Facility alarm	Facility alarm name/raising log event	Sample details string	Clearing log event
295-2430-1	tmnxPowerSupplyFanFailed	Chassis 1 Power Shelf 1 Power Module 3 fan failed	tmnxPowerSupplyFanFailed Clear
59-2004-1	linkDown	Interface intf-towards-node-B22 is not operational	linkUp
64-2091-1	tmnxSysLicenseInvalid	Error - <reason> record. <hw> will reboot the chassis <timeRemaining>	tmnxSysLicenseValid
64-2092-1	tmnxSysLicenseExpiresSoon	The license installed on <hw> expires <time Remaining>	tmnxSysLicenseValid
64-2221-1	tmnxSysStandbyLicensingError	CPM B is not licensed; license record not found	tmnxSysStandbyLicensing Ready
7-2001-1	tmnxEqCardFailure	Class MDA Module: failed, reason: Mda 1 failed startup tests	tmnxChassisNotificationClear
7-2003-1	tmnxEqCardRemoved	Class CPM Module: removed	tmnxEqCardInserted
7-2004-1	tmnxEqWrongCard	Class IOM Module: wrong type inserted	tmnxChassisNotificationClear
7-2005-1	tmnxEnvTempTooHigh	Chassis 1: temperature too high	tmnxChassisNotificationClear
7-2011-1	tmnxEqPowerSupplyRemoved	Power supply 1, power lost	tmnxEqPowerSupplyInserted
7-2017-1	tmnxEqSyncIfTimingHoldover	Synchronous Timing interface in holdover state	tmnxEqSyncIfTimingHoldover Clear
7-2019-1	tmnxEqSyncIfTimingRef1Alarm with attribute tmnxSyncIfTiming NotifyAlarm = 'los(1)'	Synchronous Timing interface, alarm los on reference 1	tmnxEqSyncIfTimingRef1Alarm Clear

Facility alarm	Facility alarm name/raising log event	Sample details string	Clearing log event
7-2019-2	tmnxEqSyncIfTimingRef1Alarm with attribute tmnxSyncIfTiming NotifyAlarm = 'oof(2)'	Synchronous Timing interface, alarm oof on reference 1	same as 7-2019-1
7-2019-3	tmnxEqSyncIfTimingRef1Alarm with attribute tmnxSyncIfTiming NotifyAlarm = 'oopir(3)'	Synchronous Timing interface, alarm oopir on reference 1	same as 7-2019-1
7-2021-x	same as 7-2019-x but for ref2	same as 7-2019-x but for ref2	same as 7-2019-x but for ref2
7-2030-x	same as 7-2019-x but for the BITS input	same as 7-2019-x but for the BITS input	same as 7-2019-x but for the BITS input
7-2033-1	tmnxChassisUpgradeInProgress	Class CPM Module: software upgrade in progress	tmnxChassisUpgradeComplete
7-2073-x	same as 7-2019-x but for the BITS2 input	same as 7-2019-x but for the BITS2 input	same as 7-2019-x but for the BITS2 input
7-2092-1	tmnxEqPowerCapacityExceeded	The system has reached maximum power capacity <x> watts	tmnxEqPowerCapacity ExceededClear
7-2094-1	tmnxEqPowerLostCapacity	The system can no longer support configured devices. Power capacity dropped to <x> watts	tmnxEqPowerLostCapacity Clear
7-2096-1	tmnxEqPowerOverloadState	The system has reached critical power capacity. Increase available power now	tmnxEqPowerOverloadState Clear
7-2104-1	tmnxEqLowSwitchFabricCap	The switch fabric capacity is less than the forwarding capacity of IOM 1 because of errors in fabric links	tmnxEqLowSwitchFabricCap Clear
7-2134-1	tmnxSyncIfTimBITS2048khz Unsup	The revision of 1/1 does not meet the specifications to support the 2048kHz BITS interface type	tmnxSyncIfTimBITS2048khz UnsupClr
7-2136-1	tmnxEqMgmtEthRedStandby Raise	The standby CPM's management Ethernet port A/1 is serving as the system's management Ethernet port	tmnxEqMgmtEthRedStandby Clear

Facility alarm	Facility alarm name/raising log event	Sample details string	Clearing log event
7-2138-1	tmnxEqPhysChassPowerSupOvrTmp	Power supply 2 over temperature	tmnxEqPhysChassPowerSupOvrTmpClr
7-2140-1	tmnxEqPhysChassPowerSupAcFail	Power supply 1 AC failure	tmnxEqPhysChassPowerSupAcFailClr
7-2142-1	tmnxEqPhysChassPowerSupDcFail	Power supply 2 DC failure	tmnxEqPhysChassPowerSupDcFailClr
7-2144-1	tmnxEqPhysChassPowerSupInFail	Power supply 1 input failure	tmnxEqPhysChassPowerSupInFailClr
7-2146-1	tmnxEqPhysChassPowerSupOutFail	Power supply 1 output failure	tmnxEqPhysChassPowerSupOutFailClr
7-2148-1	tmnxEqPhysChassisFanFailure	Fan 2 failed	tmnxEqPhysChassisFanFailureClear
7-2153-1	tmnxCpmMemSizeMismatch	The standby CPM A has a different memory size than the active B	tmnxCpmMemSizeMismatchClear
7-2156-1	tmnxPhysChassPwrSupWrgFanDir	The front to back fan direction for chassis 1 power supply 1 is not supported	tmnxPhysChassPwrSupWrgFanDirClr
7-2157-1	tmnxPhysChassPwrSupPemACRect	Chassis 1 power supply 1 ac Rec1 failed or missing	tmnxPhysChassPwrSupPemACRectClr
7-2159-1	tmnxPhysChassPwrSupInputFeed	Chassis 1 power supply 1 inputFeedA not supplying power	tmnxPhysChassPwrSupInputFeedClr
7-2161-1	tmnxEqBpEpromFail	The active CPM is no longer able to access any of backplane EPROMs because of a hardware defect	tmnxEqBpEpromFailClear
7-2163-1	tmnxEqBpEpromWarning	The active CPM is no longer to access one backplane EPROM because of a hardware defect but a redundant EPROM is present and accessible.	tmnxEqBpEpromWarningClear
7-2165-1	tmnxPhysChassisPCMInputFeed	Chassis 1 pcm 1 1 not supplying power	tmnxPhysChassisPCMInputFeedClr

Facility alarm	Facility alarm name/raising log event	Sample details string	Clearing log event
7-2190-1	tmnxPhysChassisPMOutFail	Chassis 1 Power Shelf 1 Power Module 4 output failure	tmnxPhysChassisPMOutFailClr
7-2192-1	tmnxPhysChassisPMInputFeed	Chassis 1 Power Shelf 1 Power Module 3 inputFeedA inputFeedB not supplying power	tmnxPhysChassisPMInputFeed Clr
7-2194-1	tmnxPhysChassisFilterDoorOpen	Filter door is missing or open	tmnxPhysChassisFilterDoor Closed
7-2196-1	tmnxPhysChassisPMOverTemp	Chassis 1 Power Shelf 1 over temperature	tmnxPhysChassisPMOverTemp Clr
7-2203-x	same as 7-2019-x but for SyncE	same as 7-2019-x but for SyncE	same as 7-2019-x but for SyncE
7-2205-x	same as 7-2019-x but for E2	same as 7-2019-x but for E2	same as 7-2019-x but for E2
7-4001-1	tmnxInterChassisCommsDown	Control communications disrupted between the Active CPM and the chassis	tmnxInterChassisCommsUp
7-4003-1	tmnxCpmIcPortDown	CPM Interconnect Port is not operational. Error code = invalid-connection	tmnxCpmIcPortUp
7-4006-1	tmnxCpmIcPortSFFRemoved	CPM interconnect port SFF removed	tmnxCpmIcPortSFFInserted
7-4007-1	tmnxCpmNoLocalIcPort	CPM A cannot reach the chassis using its local CPM interconnect ports	tmnxCpmLocalIcPortAvail
7-4017-1	tmnxSfmIcPortDown	SFM interconnect Port is not operational. Error code = invalid-connection to Fabric 10 IcPort 2	tmnxSfmIcPortUp
7-6002-1	tmnxPowerShelfCommsDown	Chassis 1 Power Shelf 1 lost communication with cpmA	tmnxPowerShelfCommsUp
7-6005-1	tmnxPowerShelfOutputStatus Down	Chassis 1 Power Shelf 2 output status switched to off	tmnxPowerShelfOutputStatus Up
7-6007-1	tmnxEqCardMissing	XIOM 2/x1 Class XIOM Module : missing	tmnxEqCardMissingClear

Table 59: Facility alarm name/raising log event, cause, effect and recovery

Facility alarm	Facility alarm name/ raising log event	Cause	Effect	Recovery
295-2430-1	tmnxPowerSupplyFan Failed	The tmnxPowerSupply FanFailed notification is generated when a fan within a particular power-supply has ceased to function normally.	Cooling to the power-supply may be reduced, potentially leading to overheating.	The power-supply should be replaced by one with fully-functioning fan elements.
59-2004-1	linkDown	A linkDown trap signifies that the SNMP entity, acting in an agent role, has detected that the ifOper Status object for one of its communication links is about to enter the down state from some other state (but not from the not Present state).	The indicated interface is taken down.	If the ifAdminStatus is down then the interface state is deliberate and there is no recovery. If the ifAdminStatus is up then try to determine that cause of the interface going down: cable cut, distal end went down, and so on.
64-2091-1	tmnxSysLicense Invalid	Generated when the license becomes invalid for the reason specified in the log event/alarm.	The system reboots at the end of the time remaining.	Configure a valid license file location and filename.
64-2092-1	tmnxSysLicense ExpiresSoon	Generated when the license expires soon.	The system reboots at the end of the time remaining.	Configure a valid license file location and filename.
64-2221-1	tmnxSysStandby LicensingError	Generated when the standby detects a licensing failure. The reason is specified in tmnxSys LicenseErrorReason.	The standby CPM may not synchronized and may be put into a failed state.	Configure a valid license file location and filename, given the value of tmnxSys LicenseErrorReason.
7-2001-1	tmnxEqCardFailure	Generated when one of the cards in a chassis has failed. The card type may be IOM (or XCM), MDA (or XMA), SFM, CCM, CPM, Compact Flash, and so on. The reason is indicated in the details of the log event or alarm, and also available in the tmnx ChassisNotifyCardFailure Reason attribute included in the SNMP notification.	The effect is dependent on the card that has failed. IOM (or XCM) or MDA (or XMA) failure causes a loss of service for all services running on that card. A fabric failure can impact traffic to and from all cards.	Before taking any recovery steps collect a tech-support file, then try resetting (clear) the card. If unsuccessful, try removing and re-inserting the card. If that does not work then replace the card.

Facility alarm	Facility alarm name/ raising log event	Cause	Effect	Recovery
7-2003-1	tmnxEqCardRemoved	Generated when a card is removed from the chassis. The card type may be IOM (or XCM), MDA (or XMA), SFM, CCM, CPM, Compact Flash, and so on.	The effect is dependent on the card that has been removed. IOM (or XCM) or MDA (or XMA) removal causes a loss of service for all services running on that card. A fabric removal can impact traffic to and from all cards.	Before taking any recovery steps collect a tech-support file, then try re-inserting the card. If unsuccessful, replace the card.
7-2004-1	tmnxEqWrongCard	Generated when the wrong type of card is inserted into a slot of the chassis. Even though a card may be physically supported by the slot, it may have been administratively configured to allow only specific card types in a particular slot location. The card type may be IOM (or XCM), MDA (or XMA), SFM, CCM, CPM, Compact Flash, and so on.	The effect is dependent on the card that has been incorrectly inserted. Incorrect IOM (or XCM) or MDA (or XMA) insertion causes a loss of service for all services running on that card.	Insert the correct card into the correct slot, and ensure the slot is configured for the correct type of card.
7-2005-1	tmnxEnvTempTooHigh	Generated when the temperature sensor reading on an equipment object is greater than its configured threshold.	This could be causing intermittent errors and could also cause permanent damage to components.	Remove or power off the affected cards, or improve the cooling to the node. More powerful fan trays may also be required.
7-2011-1	tmnxEqPowerSupply Removed	Generated when: • one of the power supplies is removed from the chassis • low input voltage is detected. The alarm is raised if the system detects that the voltage of the power supply has dropped to -42.5 VDC.	Reduced power can cause intermittent errors and could also cause permanent damage to components.	Re-insert the power supply or raise the input voltage to above -42.5 VDC.
7-2017-1	tmnxEqSyncIfTiming Holdover	Generated when the synchronous equipment timing subsystem transitions into a holdover state.	Any node-timed ports have very slow frequency drift limited by the central clock oscillator stability. The oscillator meets the	Address issues with the central clock input references.

Facility alarm	Facility alarm name/ raising log event	Cause	Effect	Recovery
			holdover requirements of a Stratum 3 and G.813 Option 1 clock.	
7-2019-1	tmnxEqSyncIfTiming Ref1Alarm with attribute tmnxSync IfTimingNotifyAlarm = 'los(1)'	Generated when an alarm condition on the first timing reference is detected. The type of alarm (los, oof, and so on) is indicated in the details of the log event or alarm, and is also available in the tmnx SyncIfTimingNotifyAlarm attribute included in the SNMP notification. The SNMP notification has the same indexes as those of the tmnxCpmCardTable.	Timing reference 1 cannot be used as a source of timing into the central clock.	Address issues with the signal associated with timing reference 1.
7-2019-2	tmnxEqSyncIfTiming Ref1Alarm with attribute tmnxSync IfTimingNotifyAlarm = 'oof(2)'	The same cause as 7-2019-1	The same effect as 7-2019-1	Address issues with the signal associated with timing reference 1.
7-2019-3	tmnxEqSyncIfTiming Ref1Alarm with attribute tmnxSync IfTimingNotifyAlarm = 'oopir(3)'	The same cause as 7-2019-1	The same effect as 7-2019-1	Address issues with the signal associated with timing reference 1.
7-2021-x	same as 7-2019-x but for ref2	The same cause as 7-2019-x but for the second timing reference	The same as 7-2019-x but for the second timing reference	The same as 7-2019-x but for the second timing reference
7-2030-x	same as 7-2019-x but for the BITS input	The same cause as 7-2019-x but for the BITS timing reference	The same as 7-2019-x but for the BITS timing reference	The same as 7-2019-x but for the BITS timing reference
7-2033-1	tmnxChassisUpgrade InProgress	The tmnxChassisUp grade InProgress notification is generated only after a CPM switchover occurs and the new active CPM is running new software, while the IOMs or XCMs are still running old software. This is the start of the upgrade process. The tmnxChassisUpgrade	A software mismatch between the CPM and IOM or XCM is generally fine for a short duration (during an upgrade) but may not allow for correct long term operation.	Complete the upgrade of all IOMs or XCMs.

Facility alarm	Facility alarm name/ raising log event	Cause	Effect	Recovery
		InProgress notification continues to be generated every 30 minutes while at least one IOM or XCM is still running older software.		
7-2073-x	same as 7-2019-x but for the BITS2 input	The same as 7-2019-x but for the BITS 2 timing reference	The same as 7-2019-x but for the BITS 2 timing reference	The same as 7-2019-x but for the BITS 2 timing reference
7-2092-1	tmnxEqPower CapacityExceeded	Generated when a device needs power to boot, but there is not enough power capacity to support the device.	A non-powered device does not boot until the power capacity is increased to support the device.	Add a new power supply to the system, or change the faulty power supply with a working one.
7-2094-1	tmnxEqPowerLost Capacity	Generated when a power supply fails or is removed which puts the system in an overloaded situation.	Devices are powered off in order of lowest power priority until the available power capacity can support the powered devices.	Add a new power supply to the system, or change the faulty power supply with a working one.
7-2096-1	tmnxEqPower OverloadState	Generated when the overloaded power capacity cannot support the power requirements and there are no further devices that can be powered off.	The system runs a risk of experiencing brownouts while the available power capacity does not meet the required power consumption.	Add power capacity or manually shutdown devices until the power capacity meets the power needs.
7-2104-1	tmnxEqLowSwitch FabricCap	The tmnxEqLowSwitch FabricCap alarm is generated when the total switch fabric capacity becomes less than the IOM capacity because of link failures. At least one of the taps on the IOM is below 100% capacity.	There is diminished switch fabric capacity to forward service-impacting information.	If the system does not self-recover, the IOM must be rebooted.
7-2134-1	tmnxSync IfTimBITS2048khz Unsup	The tmnxSync IfTimBITS2048khzUnsup notification is generated when the value of tSync IfTimingAdmBITSIfType is set to 'g703-2048khz (5)' and the CPM does not meet the specifications for the 2048kHz BITS output signal under G.703.	The BITS input is not used as the Sync reference and the 2048kHz BITS output signal generated by the CPM is squelched.	Replace the CPM with one that is capable of generating the 2048k Hz BITS output signal, or set tSyncIfTiming AdmBITSIfType to a value other than 'g703-2048khz (5)'.

Facility alarm	Facility alarm name/ raising log event	Cause	Effect	Recovery
7-2136-1	tmnxEqMgmtEthRed StandbyRaise	The tmnxEqMgmtEth RedStandbyRaise notification is generated when the active CPM's management Ethernet port goes operationally down and the standby CPM's management Ethernet port is operationally up and now serving as the system's management Ethernet port.	The management Ethernet port is no longer redundant. The node can be managed via the standby CPM's management Ethernet port only.	Bring the active CPM's management Ethernet port operationally up.
7-2138-1	tmnxEqPhysChass PowerSupOvrTmp	Generated when the temperature sensor reading on a power supply module is greater than its configured threshold.	This could be causing intermittent errors and could also cause permanent damage to components.	Remove or power off the affected power supply module or improve the cooling to the node. More powerful fan trays may also be required. The power supply itself may be faulty so replacement may be necessary.
7-2140-1	tmnxEqPhysChass PowerSupAcFail	Generated when an AC failure is detected on a power supply.	Reduced power can cause intermittent errors and could also cause permanent damage to components.	First try re-inserting the power supply. If unsuccessful, replace the power supply.
7-2142-1	tmnxEqPhysChass PowerSupDcFail	Generated when an DC failure is detected on a power supply.	Reduced power can cause intermittent errors and could also cause permanent damage to components.	First try re-inserting the power supply. If unsuccessful, then replace the power supply.
7-2144-1	tmnxEqPhysChass PowerSupInFail	Generated when an input failure is detected on a power supply.	Reduced power can cause intermittent errors and could also cause permanent damage to components.	First try re-inserting the power supply. If that does not work, then replace the power supply.
7-2146-1	tmnxEqPhysChass PowerSupOutFail	Generated when an output failure is detected on a power supply.	Reduced power can cause intermittent errors and could also cause permanent damage to components.	First try re-inserting the power supply. If that does not work, then replace the power supply.

Facility alarm	Facility alarm name/ raising log event	Cause	Effect	Recovery
7-2148-1	tmnxEqPhysChassis FanFailure	Generated when one of the fans in a fan tray has failed.	This could cause the temperature to rise and result in intermittent errors and potentially permanent damage to components.	Replace the fan tray immediately, improve the cooling to the node, or reduce the heat being generated in the node by removing cards or powering down the node.
7-2153-1	tmnxCpmMemSize Mismatch	A tmnxCpmMemSize Mismatch notification is generated when the RAM memory size of the standby CPM (that is, tmnx ChassisNotifyCpmCardSlot Num) is different from the active CPM (that is, tmnx ChassisNotifyHwIndex).	There is an increased risk of the memory overflow on the standby CPM during the CPM switchover.	Use CPMs with the same memory size.
7-2156-1	tmnxPhysChassPwr SupWrgFanDir	The tmnxPhysChass PwrSupWrgFanDirClr notification is generated when the airflow direction of the power supply's fan is corrected.	The fan is cooling the power supply in the correct direction.	No recovery required.
7-2157-1	tmnxPhysChassPwr SupPemACRect	The tmnxPhysChass PwrSupPemACRect notification is generated if any one of the AC rectifiers for a power supply is in a failed state or is missing.	There is an increased risk of the power supply failing, causing insufficient power to the system.	Bring the AC rectifiers back online.
7-2159-1	tmnxPhysChassPwr SupInputFeed	The tmnxPhysChassPwr SupInputFeed notification is generated if any one of the input feeds for a power supply is not supplying power.	There is an increased risk of system power brown-outs or black-outs.	Restore all of the input feeds that are not supplying power.
7-2161-1	tmnxEqBpEpromFail	The tmnxEqBpEprom Fail alarm is generated when the active CPM is no longer able to access any of backplane EPROMs because of a hardware defect.	The active CPM is at risk of failing to initialize after node reboot because of not being able to access the BP EPROM to read the chassis type.	The system does not self-recover and Nokia Support has to be contacted for further instructions.

Facility alarm	Facility alarm name/ raising log event	Cause	Effect	Recovery
7-2163-1	tmnxEqBpEprom Warning	The tmnxEqBpEprom Warning alarm is generated when the active CPM is no longer to access one backplane EPROM because of a hardware defect but a redundant EPROM is present and accessible.	There is no effect on system operation.	No recovery action required.
7-2165-1	tmnxPhys ChassisPCMInput Feed	The tmnxPhys ChassisPCMInputFeed notification is generated if any one of the input feeds for a PCM has gone offline.	There is an increased risk of system power brown-outs or black-outs.	Restore all of the input feeds that are not supplying power.
7-2190-1	tmnxPhys ChassisPMOutFail	The tmnxPhys ChassisPMOutFail notification is generated when an output failure occurs on the power module.	The power module is no longer operational.	Insert a new power module.
7-2192-1	tmnxPhys ChassisPMInputFeed	The tmnxPhys ChassisPMInputFeed notification is generated if any one of the input feeds for a power module is not supplying power.	There is an increased risk of system power brownouts or blackouts.	Restore all of the input feeds that are not supplying power.
7-2194-1	tmnxPhysChassis FilterDoorOpen	The tmnxPhysChassisFilter DoorOpen notification is generated when the filter door is either open or not present.	Power shelf protection may be compromised.	If the filter door is not installed, install it. Close the filter door.
7-2196-1	tmnxPhys ChassisPMOverTemp	The tmnxPhys ChassisPMOverTemp notification is generated when a power module's temperature surpasses the temperature threshold.	The power module is no longer operational.	Check input feed or insert a new power module.
7-2203-x	same as 7-2019-x but for SyncE	The same cause as 7-2019-x but for SyncE	same as 7-2019-x but for SyncE	same as 7-2019-x but for SyncE
7-2205-x	same as 7-2019-x but for E2	The same cause as 7-2019-x but for E2	same as 7-2019-x but for E2	same as 7-2019-x but for E2

Facility alarm	Facility alarm name/ raising log event	Cause	Effect	Recovery
7-4001-1	tmnxInterChassis CommsDown	The tmnxInterChassis CommsDown alarm is generated when the active CPM cannot reach the far-end chassis.	The resources on the far-end chassis are not available. This event for the far-end chassis means that the CPM, SFM, and XCM cards in the far-end chassis reboot and remain operationally down until communications are re-established.	Ensure that all CPM interconnect ports in the system are properly cabled together with working cables.
7-4003-1	tmnxCpmIcPortDown	The tmnxCpmIcPort Down alarm is generated when the CPM interconnect port is not operational. The reason may be a cable connected incorrectly, a disconnected cable, a faulty cable, or a misbehaving CPM interconnect port or card.	At least one of the control plane paths used for inter-chassis CPM communication is not operational. Other paths may be available.	A manual verification and testing of each CPM interconnect port is required to ensure fully functional operation. Physical replacement of cabling may be required.
7-4006-1	tmnxCpm IcPortSFFRemoved	The tmnxCpm IcPortSFFRemoved notification is generated when the SFF (eg. QSFP) is removed from the CPM interconnect port. Removing an SFF causes both this trap, and also a tmnxCpmIcPortDown event.	Removing the SFF causes the CPM interconnect port to go down. This port is no longer able to be used as part of the control plane between chassis but other paths may be available.	Insert a working SFF into the port.
7-4007-1	tmnxCpmNoLocal IcPort	The tmnxCpmNoLo cal IcPort alarm is generated when the CPM cannot reach the other chassis using its local CPM interconnect ports.	Another control communications path may still be available between the CPM and the other chassis via the mate CPM in the same chassis. If that alternative path is not available then complete disruption of control communications to the other chassis occurs and the tmnxInterChassis CommsDown alarm is raised. A tmnxCpmNoLocalIcPort alarm on the active CPM	Ensure that all CPM interconnect ports in the system are properly cabled together with working cables.

Facility alarm	Facility alarm name/ raising log event	Cause	Effect	Recovery
			indicates that a further failure of the local CPM interconnect ports on the standby CPM causes complete disruption of control communications to the other chassis and the tmnxInterChassisComms Down alarm is raised. A tmnxCpmNoLocalIcPort alarm on the standby CPM indicates that a CPM switchover may cause temporary disruption of control communications to the other chassis while the rebooting CPM comes back into service.	
7-4017-1	tmnxSfmIcPortDown	The tmnxSfmIcPortDown alarm is generated when the SFM interconnect port is not operational. The reason may be a cable connected incorrectly, a disconnected cable, a faulty cable, or a misbehaving SFM interconnect port or SFM card.	This port can no longer be used as part of the user plane fabric between chassis. Other fabric paths may be available resulting in no loss of capacity.	A manual verification and testing of each SFM interconnect port is required to ensure fully functional operation. Physical replacement of cabling may be required.
7-6002-1	tmnxPowerShelf CommsDown	The tmnxPowerShelf CommsDown is generated when there is a loss of communications with the power shelf controller.	If there is a power failure, it is not detected because the power modules cannot be polled. The system continues to report the state of the power modules as they were when last seen.	Correct the power shelf controller communications problem.
7-6005-1	tmnxPowerShelf OutputStatusDown	The tmnxPowerShelf OutputStatusSwitch is generated when the physical output switch on the power shelf is set to Standby.	The power output from the identified power shelf is switched off and does not supply power to the system.	Set output switch to On to restore power output.
7-6007-1	tmnxEqCardMissing	A provisioned and administratively enabled	A card that is expected by the user to be present	Investigate the physical equipment to identify

Facility alarm	Facility alarm name/ raising log event	Cause	Effect	Recovery
		card in the specific slot has failed to be identified on system bootup or CPM switchover. This is reported 15 minutes after the active CPM completes its boot process to allow for repeated card boot attempts.	in a system has not been identified. It will have either been removed during or before bootup or has failed and is not recognized as inserted.	whether the card is missing. If the card is present attempt to reseat the card. If this is unsuccessful contact Nokia technical support.

Table 60: ESA facility alarm, facility alarm name, raising log event, sample details string, and clearing log event and Table 61: ESA facility alarm name/raising log event, cause, effect, and recovery show the supported ESA facility alarms.

Table 60: ESA facility alarm, facility alarm name, raising log event, sample details string, and clearing log event

ESA Event and states	Facility Alarm	Severity	Facility alarm name/raising log event	Sample details string	Clearing log event
ESA HW Status: degraded	2400-1	Major	tmnxEsaHwStatusDegraded	ESA 3 aggregate hardware status degraded	tmnxEsaHwStatusDegraded Clr
ESA HW Status: critical	2402-1	Critical	tmnxEsaHwStatusCritical	ESA 3 aggregate hardware status critical	tmnxEsaHwStatusCriticalClr
ESA HW Power Supply 1 Degraded	2404-1	Critical	tmnxEsaHwPwr Sup1Degraded	ESA 3 power supply 1 status degraded	tmnxEsaHwPwr Sup1DegradedClr
ESA HW Power Supply 1 Failed	2406-1	Critical	tmnxEsaHwPwrSup1Failed	ESA 3 power supply 1 status failed	tmnxEsaHwPwrSup1Failed Clr
ESA HW Power Supply 2 Degraded	2408-1	Critical	tmnxEsaHwPwr Sup2Degraded	ESA 3 power supply 2 status degraded	tmnxEsaHwPwr Sup2DegradedClr
ESA HW Power Supply 2 Failed	2410-1	Critical	tmnxEsaHwPwrSup2Failed	ESA 3 power supply 2 status failed	tmnxEsaHwPwrSup2Failed Clr
ESA HW Fan Bank Non Redundant	2412-1	Major	tmnxEsaHwFanBankNon Redun	ESA 3 fan bank	tmnxEsaHwFanBankNon RedunClr

ESA Event and states	Facility Alarm	Severity	Facility alarm name/raising log event	Sample details string	Clearing log event
				redundancy degraded	
ESA HW Fan Bank Failed Redundancy	2414-1	Critical	tmnxEsaHwFanBankFail Redun	ESA 3 fan bank redundancy failed	tmnxEsaHwFanBankFail RedunClr
ESA HW Fan Status Degraded	2416-1	Critical	tmnxEsaHwFanStatus Degraded	ESA 3 fan status degraded	tmnxEsaHwFanStatus DegradedClr
ESA HW Fan Status Failed	2418-1	Critical	tmnxEsaHwFanStatusFailed	ESA 3 fan status failed	tmnxEsaHwFanStatusFailed Clr
ESA HW Power Supply Mismatch	2420-1	Major	tmnxEsaHwPwrSup Mismatch	ESA 3 power supply mismatch	tmnxEsaHwPwrSup MismatchClr
ESA HW Power Supply Bank Non Redundant	2422-1	Major	tmnxEsaHwPwrSupBank NonRedun	ESA 3 power supply bank redundancy degraded	tmnxEsaHwPwrSupBank NonRedunClr
ESA HW Temperature Degraded	2426-1	Critical	tmnxEsaHwTemperature Degraded	ESA 3 temperature status degraded	tmnxEsaHwTemperature DegradedClr
ESA HW Temperature Failed	2428-1	Critical	tmnxEsaHwTemperature Failed	ESA 3 temperature status failed	tmnxEsaHwTemperature FailedClr

Table 61: ESA facility alarm name/raising log event, cause, effect, and recovery

Facility Alarm	Facility alarm name/raising log event	Cause	Effect	Recovery
2400-1	tmnxEsaHwStatusDegraded	Generated when the ESA hardware status is degraded	Service may be affected	Contact Nokia customer support
2402-1	tmnxEsaHwStatusCritical	Generated when one or more ESA hardware statuses are critical	Service may be affected	Contact Nokia customer support
2404-1	tmnxEsaHwPwr Sup1Degraded	Generated when the ESA power supply 1 is degraded	Power supply and redundancy are affected. ESA operation may be affected.	Contact Nokia customer support

Facility Alarm	Facility alarm name/raising log event	Cause	Effect	Recovery
2406-1	tmnxEsaHwPwrSup1Failed	Generated when the ESA power supply 1 fails	Power supply and redundancy are affected. ESA operation may be affected.	Contact Nokia customer support
2408-1	tmnxEsaHwPwr Sup2Degraded	Generated when the ESA power supply 2 is degraded	Power supply operation and reliability may be affected	Contact Nokia customer support
2410-1	tmnxEsaHwPwrSup2Failed	Generated when the ESA power supply 2 fails	Power supply and redundancy are affected. ESA operation may be affected.	Contact Nokia customer support
2412-1	tmnxEsaHwFanBankNon Redun	Generated when 1 to 6 of the 7 fans are failed	ESA cooling may be inadequate	Contact Nokia customer support
2414-1	tmnxEsaHwFanBankFail Redun	Generated when all 7 fans are failed	ESA cooling is inadequate, the ESA may shut down	Contact Nokia customer support
2416-1	tmnxEsaHwFanStatus Degraded	Generated when one or more ESA fans are degraded	ESA cooling may be inadequate	Contact Nokia customer support
2418-1	tmnxEsaHwFanStatusFailed	Generated when one or more ESA fans fail	ESA cooling may be inadequate	Contact Nokia customer support
2420-1	tmnxEsaHwPwrSupMismatch	Generated when the ESA power supplies do not match	ESA power supplies must be matched	Equip the ESA with matching power supplies
2422-1	tmnxEsaHwPwrSupBankNon Redun	Generated when one ESA power supply has failed and the other is running	ESA will fail if another power supply failure occurs	Contact Nokia customer support
2426-1	tmnxEsaHwTemperature Degraded	Generated when one or more ESA temperatures is outside the expected operating range	If the ESA temperature remains outside the expected operating range,	The ESA may need maintenance to rectify the issue. Contact Nokia customer support.

Facility Alarm	Facility alarm name/raising log event	Cause	Effect	Recovery
			the ESA may shut down	
2428-1	tmnxEsaHwTemperature Failed	Generated when the ESA temperature is critical	If the ESA temperature remains outside the expected operating range, the ESA may shut down	The ESA may need maintenance to rectify the issue. Contact Nokia customer support.

The linkDown Facility Alarm is supported for the objects listed in Table 62: linkDown Facility Alarm support (note that all objects may not be supported on all platforms):

Table 62: linkDown Facility Alarm support

Object	Supported
Ethernet Ports	Yes
Sonet Section, Line and Path (POS)	Yes
TDM Ports (E1, T1, DS3) including CES MDAs	Yes
TDM Channels (DS3 channel configured in an STM-1 port)	Yes
Ethernet LAGs	No
APS groups	No
Ethernet VLANs	No

13.6 Configuring facility alarms with CLI

This section provides information to configure facility alarms using the CLI.

13.6.1 Basic facility alarm configuration

The following example shows the enabling of facility alarms.

Example: MD-CLI

```
[ex:/configure system alarms]
A:admin@node-2# info
    admin-state enable
```

Example: classic CLI

```
A:node-2>config>system>alarms# info detail
----------------------------------------------
            no shutdown
            exit
----------------------------------------------
```

13.6.2 Common configuration tasks

The following sections are basic alarm tasks that can be performed.

13.6.2.1 Configuring the maximum number of alarms to clear

The number of alarms to clear can be configured using the following command.

```
configure system alarms max-cleared
```

Example: MD-CLI

```
[ex:/configure system alarms]
A:admin@node-2# info
    max-cleared 100
```

Example: classic CLI

```
A:node-2>config>system# alarms
----------------------------------------------
...
max-cleared 100
exit
...
----------------------------------------------
```

14 Standards and protocol support

Note:
The information provided in this chapter is subject to change without notice and may not apply to all platforms.

Nokia assumes no responsibility for inaccuracies.

14.1 Bidirectional Forwarding Detection (BFD)

RFC 5880, *Bidirectional Forwarding Detection (BFD)*

RFC 5881, *Bidirectional Forwarding Detection (BFD) IPv4 and IPv6 (Single Hop)*

RFC 5882, *Generic Application of Bidirectional Forwarding Detection (BFD)*

14.2 Border Gateway Protocol (BGP)

draft-hares-idr-update-attrib-low-bits-fix-01, *Update Attribute Flag Low Bits Clarification*

draft-ietf-idr-add-paths-guidelines-08, *Best Practices for Advertisement of Multiple Paths in IBGP*

draft-ietf-idr-best-external-03, *Advertisement of the best external route in BGP*

draft-ietf-idr-bgp-gr-notification-01, *Notification Message support for BGP Graceful Restart*

draft-ietf-idr-bgp-optimal-route-reflection-10, *BGP Optimal Route Reflection (BGP-ORR)*

draft-ietf-idr-error-handling-03, *Revised Error Handling for BGP UPDATE Messages*

draft-ietf-idr-link-bandwidth-03, *BGP Link Bandwidth Extended Community*

RFC 1772, *Application of the Border Gateway Protocol in the Internet*

RFC 1997, *BGP Communities Attribute*

RFC 2385, *Protection of BGP Sessions via the TCP MD5 Signature Option*

RFC 2439, *BGP Route Flap Damping*

RFC 2545, *Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing*

RFC 2858, *Multiprotocol Extensions for BGP-4*

RFC 2918, *Route Refresh Capability for BGP-4*

RFC 4271, *A Border Gateway Protocol 4 (BGP-4)*

RFC 4360, *BGP Extended Communities Attribute*

RFC 4364, *BGP/MPLS IP Virtual Private Networks (VPNs)*

RFC 4456, *BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)*

RFC 4486, *Subcodes for BGP Cease Notification Message*

RFC 4659, *BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN*

RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*

RFC 4724, *Graceful Restart Mechanism for BGP* – helper mode

RFC 4760, *Multiprotocol Extensions for BGP-4*

RFC 4798, *Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)*

RFC 5004, *Avoid BGP Best Path Transitions from One External to Another*

RFC 5065, *Autonomous System Confederations for BGP*

RFC 5291, *Outbound Route Filtering Capability for BGP-4*

RFC 5396, *Textual Representation of Autonomous System (AS) Numbers* – asplain

RFC 5492, *Capabilities Advertisement with BGP-4*

RFC 5668, *4-Octet AS Specific BGP Extended Community*

RFC 6286, *Autonomous-System-Wide Unique BGP Identifier for BGP-4*

RFC 6368, *Internal BGP as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)*

RFC 6793, *BGP Support for Four-Octet Autonomous System (AS) Number Space*

RFC 6810, *The Resource Public Key Infrastructure (RPKI) to Router Protocol*

RFC 6811, *Prefix Origin Validation*

RFC 6996, *Autonomous System (AS) Reservation for Private Use*

RFC 7311, *The Accumulated IGP Metric Attribute for BGP*

RFC 7606, *Revised Error Handling for BGP UPDATE Messages*

RFC 7607, *Codification of AS 0 Processing*

RFC 7752, *North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP*

RFC 7911, *Advertisement of Multiple Paths in BGP*

RFC 7999, *BLACKHOLE Community*

RFC 8092, *BGP Large Communities Attribute*

RFC 8097, *BGP Prefix Origin Validation State Extended Community*

RFC 8212, *Default External BGP (EBGP) Route Propagation Behavior without Policies*

RFC 8277, *Using BGP to Bind MPLS Labels to Address Prefixes*

RFC 9294, *Application-Specific Link Attributes Advertisement Using the Border Gateway Protocol - Link State (BGP LS)*

RFC 9494, *Long-Lived Graceful Restart for BGP*

14.3 Bridging and management

IEEE 802.1AB, *Station and Media Access Control Connectivity Discovery*

IEEE 802.1ad, *Provider Bridges*

IEEE 802.1AX, *Link Aggregation*

IEEE 802.1D, *MAC Bridges*

IEEE 802.1p, *Traffic Class Expediting*

IEEE 802.1Q, *Virtual LANs*

IEEE 802.1s, *Multiple Spanning Trees*

IEEE 802.1w, *Rapid Reconfiguration of Spanning Tree*

14.4 Certificate management

RFC 4210, *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)*

RFC 4211, *Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)*

RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

RFC 6712, *Internet X.509 Public Key Infrastructure -- HTTP Transfer for the Certificate Management Protocol (CMP)*

RFC 7030, *Enrollment over Secure Transport*

RFC 7468, *Textual Encodings of PKIX, PKCS, and CMS Structures*

14.5 Ethernet

IEEE 802.3x, *Ethernet Flow Control*

14.6 Ethernet VPN (EVPN)

draft-ietf-bess-evpn-ipvpn-interworking-15, *EVPN Interworking with IPVPN*

draft-ietf-bess-evpn-l3mh-proto-00, *EVPN Multi-Homing support for L3 services*

draft-rbickhart-evpn-ip-mac-proxy-adv-04, *Proxy MAC-IP Advertisement in EVPN*

RFC 7432, *BGP MPLS-Based Ethernet VPN*

RFC 8214, *Virtual Private Wire Service Support in Ethernet VPN*

RFC 8365, *A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)*

RFC 8560, *Seamless Integration of Ethernet VPN (EVPN) with Virtual Private LAN Service (VPLS) and Their Provider Backbone Bridge (PBB) Equivalents*

RFC 9047, *Propagation of ARP/ND Flags in an Ethernet Virtual Private Network (EVPN)*

RFC 9135, *Integrated Routing and Bridging in Ethernet VPN (EVPN)*

RFC 9136, *IP Prefix Advertisement in Ethernet VPN (EVPN)*

RFC 9161, *Operational Aspects of Proxy ARP/ND in Ethernet Virtual Private Networks*

RFC 9251, *Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Proxies for Ethernet VPN (EVPN)*

14.7 gRPC Remote Procedure Calls (gRPC)

cert.proto version 0.1.0, *gNOI Certificate Management Service*

file.proto version 0.1.0, *gNOI File Service*

gnmi.proto version 0.8.0, *gNMI Service Specification*

gnmi_ext.proto, *gNMI Commit Confirmed Extension*

gnmi_ext.proto, *gNMI Config Subscription Extension*

gnmi_ext.proto, *gNMI Depth Extension*

system.proto version 1.0.0, *gNOI System Service*

tunnel.proto version 0.2, *gRPC Tunnel Service*

PROTOCOL-HTTP2, *gRPC over HTTP2*

14.8 Intermediate System to Intermediate System (IS-IS)

draft-ietf-isis-mi-02, *IS-IS Multi-Instance*

draft-ietf-lsr-igp-ureach-prefix-announce-01, *IGP Unreachable Prefix Announcement* – without U-Flag and UP-Flag

draft-kaplan-isis-ext-eth-02, *Extended Ethernet Frame Size Support*

ISO/IEC 10589:2002 Second Edition, *Intermediate system to Intermediate system intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473)*

RFC 1195, *Use of OSI IS-IS for Routing in TCP/IP and Dual Environments*

RFC 2973, *IS-IS Mesh Groups*

RFC 3359, *Reserved Type, Length and Value (TLV) Codepoints in Intermediate System to Intermediate System*

RFC 3719, *Recommendations for Interoperable Networks using Intermediate System to Intermediate System (IS-IS)*

RFC 3787, *Recommendations for Interoperable IP Networks using Intermediate System to Intermediate System (IS-IS)*

RFC 5120, *M-ISIS: Multi Topology (MT) Routing in IS-IS*

RFC 5130, *A Policy Control Mechanism in IS-IS Using Administrative Tags*

RFC 5301, *Dynamic Hostname Exchange Mechanism for IS-IS*

RFC 5302, *Domain-wide Prefix Distribution with Two-Level IS-IS*

RFC 5303, *Three-Way Handshake for IS-IS Point-to-Point Adjacencies*

RFC 5304, *IS-IS Cryptographic Authentication*

RFC 5305, *IS-IS Extensions for Traffic Engineering TE*

RFC 5306, *Restart Signaling for IS-IS* – helper mode

RFC 5308, *Routing IPv6 with IS-IS*

RFC 5309, *Point-to-Point Operation over LAN in Link State Routing Protocols*

RFC 5310, *IS-IS Generic Cryptographic Authentication*

RFC 6213, *IS-IS BFD-Enabled TLV*

RFC 6232, *Purge Originator Identification TLV for IS-IS*

RFC 6233, *IS-IS Registry Extension for Purges*

RFC 7775, *IS-IS Route Preference for Extended IP and IPv6 Reachability*

RFC 7794, *IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability* – sections 2.1 and 2.3

RFC 7981, *IS-IS Extensions for Advertising Router Information*

RFC 7987, *IS-IS Minimum Remaining Lifetime*

RFC 8202, *IS-IS Multi-Instance* – single topology

RFC 8570, *IS-IS Traffic Engineering (TE) Metric Extensions* – Min/Max Unidirectional Link Delay metric for flex-algo, RSVP, SR-TE

RFC 8919, *IS-IS Application-Specific Link Attributes*

RFC 9885, *Multi-Part TLVs in IS-IS*

14.9 Internet Protocol (IP) general

RFC 768, *User Datagram Protocol*

RFC 793, *Transmission Control Protocol*

RFC 854, *Telnet Protocol Specifications*

RFC 1350, *The TFTP Protocol (revision 2)*

RFC 2784, *Generic Routing Encapsulation (GRE)*

RFC 3164, *The BSD syslog Protocol*

RFC 4250, *The Secure Shell (SSH) Protocol Assigned Numbers*

RFC 4251, *The Secure Shell (SSH) Protocol Architecture*

RFC 4252, *The Secure Shell (SSH) Authentication Protocol* – publickey, password

RFC 4253, *The Secure Shell (SSH) Transport Layer Protocol*

RFC 4254, *The Secure Shell (SSH) Connection Protocol*

RFC 4511, *Lightweight Directory Access Protocol (LDAP): The Protocol*

RFC 4513, *Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms* – TLS

RFC 4632, *Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*

RFC 5082, *The Generalized TTL Security Mechanism (GTSM)*

RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2* – TLS client, RSA public key

RFC 5289, *TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)*

RFC 5425, *Transport Layer Security (TLS) Transport Mapping for Syslog* – RFC 3164 with TLS

RFC 5656, *Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer* – ECDSA

RFC 5925, *The TCP Authentication Option*

RFC 5926, *Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)*

RFC 6398, *IP Router Alert Considerations and Usage* – MLD

RFC 6528, *Defending against Sequence Number Attacks*

RFC 8446, *The Transport Layer Security (TLS) Protocol Version 1.3*

RFC 8907, *The Terminal Access Controller Access-Control System Plus (TACACS+) Protocol*

14.10 Internet Protocol (IP) multicast

RFC 1112, *Host Extensions for IP Multicasting*

RFC 2236, *Internet Group Management Protocol, Version 2*

RFC 2365, *Administratively Scoped IP Multicast*

RFC 2375, *IPv6 Multicast Address Assignments*

RFC 2710, *Multicast Listener Discovery (MLD) for IPv6*

RFC 3376, *Internet Group Management Protocol, Version 3*

RFC 3446, *Anycast Rendevous Point (RP) mechanism using Protocol Independent Multicast (PIM) and Multicast Source Discovery Protocol (MSDP)*

RFC 3590, *Source Address Selection for the Multicast Listener Discovery (MLD) Protocol*

RFC 3618, *Multicast Source Discovery Protocol (MSDP)*

RFC 3810, *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*

RFC 4541, *Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches*

RFC 4604, *Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast*

RFC 4610, *Anycast-RP Using Protocol Independent Multicast (PIM)*

RFC 4611, *Multicast Source Discovery Protocol (MSDP) Deployment Scenarios*

RFC 5059, *Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM)*

RFC 5186, *Internet Group Management Protocol Version 3 (IGMPv3) / Multicast Listener Discovery Version 2 (MLDv2) and Multicast Routing Protocol Interaction*

RFC 5384, *The Protocol Independent Multicast (PIM) Join Attribute Format*

RFC 7761, *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)*

RFC 8487, *Mtrace Version 2: Traceroute Facility for IP Multicast*

14.11 Internet Protocol (IP) version 4

RFC 791, *Internet Protocol*

RFC 792, *Internet Control Message Protocol*

RFC 826, *An Ethernet Address Resolution Protocol*

RFC 1034, *Domain Names - Concepts and Facilities*

RFC 1035, *Domain Names - Implementation and Specification*

RFC 1191, *Path MTU Discovery* – router specification

RFC 1519, *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*

RFC 1812, *Requirements for IPv4 Routers*

RFC 1918, *Address Allocation for Private Internets*

RFC 2131, *Dynamic Host Configuration Protocol*; Relay only

RFC 2132, *DHCP Options and BOOTP Vendor Extensions* – DHCP

RFC 2401, *Security Architecture for Internet Protocol*

RFC 3021, *Using 31-Bit Prefixes on IPv4 Point-to-Point Links*

RFC 3046, *DHCP Relay Agent Information Option (Option 82)*

RFC 3768, *Virtual Router Redundancy Protocol (VRRP)*

RFC 4884, *Extended ICMP to Support Multi-Part Messages* – ICMPv4 and ICMPv6 Time Exceeded

14.12 Internet Protocol (IP) version 6

RFC 2464, *Transmission of IPv6 Packets over Ethernet Networks*

RFC 2529, *Transmission of IPv6 over IPv4 Domains without Explicit Tunnels*

RFC 3122, *Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification*

RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*

RFC 3587, *IPv6 Global Unicast Address Format*

RFC 3596, *DNS Extensions to Support IP version 6*

RFC 3633, *IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6*

RFC 3736, *Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6*

RFC 3971, *SEcure Neighbor Discovery (SEND)*

RFC 4007, *IPv6 Scoped Address Architecture*

RFC 4191, *Default Router Preferences and More-Specific Routes* – Default Router Preference

RFC 4193, *Unique Local IPv6 Unicast Addresses*

RFC 4291, *Internet Protocol Version 6 (IPv6) Addressing Architecture*

RFC 4443, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*

RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*

RFC 5095, *Deprecation of Type 0 Routing Headers in IPv6*

RFC 5722, *Handling of Overlapping IPv6 Fragments*

RFC 5798, *Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6* – IPv6

RFC 5952, *A Recommendation for IPv6 Address Text Representation*

RFC 6164, *Using 127-Bit IPv6 Prefixes on Inter-Router Links*

RFC 8021, *Generation of IPv6 Atomic Fragments Considered Harmful*

RFC 8200, *Internet Protocol, Version 6 (IPv6) Specification*

14.13 Internet Protocol Security (IPsec)

draft-ietf-ipsec-isakmp-mode-cfg-05, *The ISAKMP Configuration Method*

draft-ietf-ipsec-isakmp-xauth-06, *Extended Authentication within ISAKMP/Oakley (XAUTH)*

RFC 2401, *Security Architecture for the Internet Protocol*

RFC 2403, *The Use of HMAC-MD5-96 within ESP and AH*

RFC 2404, *The Use of HMAC-SHA-1-96 within ESP and AH*

RFC 2405, *The ESP DES-CBC Cipher Algorithm With Explicit IV*

RFC 2406, *IP Encapsulating Security Payload (ESP)*

RFC 2407, *IPsec Domain of Interpretation for ISAKMP (IPsec DoI)*

RFC 2408, *Internet Security Association and Key Management Protocol (ISAKMP)*

RFC 2409, *The Internet Key Exchange (IKE)*

RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec*

RFC 2560, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*

RFC 3526, *More Modular Exponential (MODP) Diffie-Hellman group for Internet Key Exchange (IKE)*

RFC 3566, *The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec*

RFC 3602, *The AES-CBC Cipher Algorithm and Its Use with IPsec*

RFC 3706, *A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers*

RFC 3947, *Negotiation of NAT-Traversal in the IKE*

RFC 3948, *UDP Encapsulation of IPsec ESP Packets*

RFC 4106, *The Use of Galois/Counter Mode (GCM) in IPsec ESP*

RFC 4109, *Algorithms for Internet Key Exchange version 1 (IKEv1)*

RFC 4301, *Security Architecture for the Internet Protocol*

RFC 4303, *IP Encapsulating Security Payload*

RFC 4307, *Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)*

RFC 4308, *Cryptographic Suites for IPsec*

RFC 4434, *The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)*

RFC 4543, *The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH*

RFC 4754, *IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)*

RFC 4835, *Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)*

RFC 4868, *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec*

RFC 4945, *The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2 and PKIX*

RFC 5019, *The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments*

RFC 5282, *Using Authenticated Encryption Algorithms with the Encrypted Payload of the IKEv2 Protocol*

RFC 5903, *ECP Groups for IKE and IKEv2*

RFC 5996, *Internet Key Exchange Protocol Version 2 (IKEv2)*

RFC 5998, *An Extension for EAP-Only Authentication in IKEv2*

RFC 6379, *Suite B Cryptographic Suites for IPsec*

RFC 6380, *Suite B Profile for Internet Protocol Security (IPsec)*

RFC 6960, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*

RFC 7296, *Internet Key Exchange Protocol Version 2 (IKEv2)*

RFC 7321, *Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)*

RFC 7383, *Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation*

RFC 7427, *Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)*

RFC 8784, *Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security*

14.14 Label Distribution Protocol (LDP)

draft-pdutta-mpls-ldp-adj-capability-00, *LDP Adjacency Capabilities*

draft-pdutta-mpls-ldp-v2-00, *LDP Version 2*

RFC 3037, *LDP Applicability*

RFC 3478, *Graceful Restart Mechanism for Label Distribution Protocol* – helper mode

RFC 5036, *LDP Specification*

RFC 5283, *LDP Extension for Inter-Area Label Switched Paths (LSPs)*

RFC 5443, *LDP IGP Synchronization*

RFC 5561, *LDP Capabilities*

RFC 5919, *Signaling LDP Label Advertisement Completion*

14.15 Multiprotocol Label Switching (MPLS)

RFC 3031, *Multiprotocol Label Switching Architecture*

RFC 3032, *MPLS Label Stack Encoding*

RFC 3270, *Multi-Protocol Label Switching (MPLS) Support of Differentiated Services* – E-LSP

RFC 3443, *Time To Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks*

RFC 5332, *MPLS Multicast Encapsulations*

RFC 5884, *Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)*

RFC 6424, *Mechanism for Performing Label Switched Path Ping (LSP Ping) over MPLS Tunnels*

RFC 7308, *Extended Administrative Groups in MPLS Traffic Engineering (MPLS-TE)*

RFC 7746, *Label Switched Path (LSP) Self-Ping*

14.16 Network Address Translation (NAT)

RFC 4787, *Network Address Translation (NAT) Behavioral Requirements for Unicast UDP*

RFC 5382, *NAT Behavioral Requirements for TCP*

RFC 5508, *NAT Behavioral Requirements for ICMP*

14.17 Network Configuration Protocol (NETCONF)

RFC 5277, *NETCONF Event Notifications*

RFC 6020, *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*

RFC 6022, *YANG Module for NETCONF Monitoring*

RFC 6241, *Network Configuration Protocol (NETCONF)*

RFC 6242, *Using the NETCONF Protocol over Secure Shell (SSH)*

RFC 6243, *With-defaults Capability for NETCONF*

RFC 8071, *NETCONF Call Home and RESTCONF Call Home* – NETCONF

RFC 8342, *Network Management Datastore Architecture (NMDA)* – Startup, Candidate, Running and Intended datastores

RFC 8525, *YANG Library*

RFC 8526, *NETCONF Extensions to Support the Network Management Datastore Architecture* – <get-data> operation

14.18 Media sanitization

NIST Special Publication 800-88 Revision 1, *Guidelines for Media Sanitization* – CF, MMC, SSD, SD, USB

14.19 Open Shortest Path First (OSPF)

RFC 1765, *OSPF Database Overflow*

RFC 2328, *OSPF Version 2*

RFC 3101, *The OSPF Not-So-Stubby Area (NSSA) Option*

RFC 3509, *Alternative Implementations of OSPF Area Border Routers*

RFC 3623, *Graceful OSPF Restart Graceful OSPF Restart* – helper mode

RFC 3630, *Traffic Engineering (TE) Extensions to OSPF Version 2*

RFC 4811, *OSPF Out-of-Band Link State Database (LSDB) Resynchronization* – OSPFv2

RFC 4812, *OSPF Restart Signaling* – OSPFv2

RFC 4576, *Using a Link State Advertisement (LSA) Options Bit to Prevent Looping in BGP/MPLS IP Virtual Private Networks (VPNs)*

RFC 4577, *OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)*

RFC 5185, *OSPF Multi-Area Adjacency*

RFC 5243, *OSPF Database Exchange Summary List Optimization*

RFC 5250, *The OSPF Opaque LSA Option*

RFC 5309, *Point-to-Point Operation over LAN in Link State Routing Protocols*

RFC 5642, *Dynamic Hostname Exchange Mechanism for OSPF*

RFC 6549, *OSPFv2 Multi-Instance Extensions*

RFC 6987, *OSPF Stub Router Advertisement*

RFC 7471, *OSPF Traffic Engineering (TE) Metric Extensions* – Min/Max Unidirectional Link Delay metric for flex-algo, RSVP, SR-TE

RFC 7684, *OSPFv2 Prefix/Link Attribute Advertisement*

RFC 7770, *Extensions to OSPF for Advertising Optional Router Capabilities*

RFC 8920, *OSPF Application-Specific Link Attributes*

14.20 Path Computation Element Protocol (PCEP)

draft-alvarez-pce-path-profiles-04, *PCE Path Profiles*

draft-ietf-pce-binding-label-sid-15, *Carrying Binding Label/Segment Identifier (SID) in PCE-based Networks.* – MPLS binding SIDs

RFC 5440, *Path Computation Element (PCE) Communication Protocol (PCEP)*

RFC 8231, *Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE*

RFC 8253, *PCEPS: Usage of TLS to Provide a Secure Transport for the Path Computation Element Communication Protocol (PCEP)*

RFC 8281, *PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model*

RFC 8408, *Conveying Path Setup Type in PCE Communication Protocol (PCEP) Messages*

RFC 8664, *Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing*

14.21 Pseudowire (PW)

draft-ietf-l2vpn-vpws-iw-oam-04, *OAM Procedures for VPWS Interworking*

MFA Forum 12.0.0, *Multiservice Interworking - Ethernet over MPLS*

MFA Forum 13.0.0, *Fault Management for Multiservice Interworking v1.0*

MFA Forum 16.0.0, *Multiservice Interworking - IP over MPLS*

RFC 3916, *Requirements for Pseudo-Wire Emulation Edge-to-Edge (PWE3)*

RFC 3985, *Pseudo Wire Emulation Edge-to-Edge (PWE3)*

RFC 4385, *Pseudo Wire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN*

RFC 4446, *IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)*

RFC 4447, *Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*

RFC 4448, *Encapsulation Methods for Transport of Ethernet over MPLS Networks*

RFC 5085, *Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires*

RFC 5659, *An Architecture for Multi-Segment Pseudowire Emulation Edge-to-Edge*

RFC 5885, *Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)*

RFC 6073, *Segmented Pseudowire*

RFC 6310, *Pseudowire (PW) Operations, Administration, and Maintenance (OAM) Message Mapping*

RFC 6391, *Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network*

RFC 6575, *Address Resolution Protocol (ARP) Mediation for IP Interworking of Layer 2 VPNs*

RFC 6718, *Pseudowire Redundancy*

RFC 6829, *Label Switched Path (LSP) Ping for Pseudowire Forwarding Equivalence Classes (FECs) Advertised over IPv6*

RFC 6870, *Pseudowire Preferential Forwarding Status bit*

RFC 7023, *MPLS and Ethernet Operations, Administration, and Maintenance (OAM) Interworking*

RFC 7267, *Dynamic Placement of Multi-Segment Pseudowires*

RFC 7392, *Explicit Path Routing for Dynamic Multi-Segment Pseudowires* – ER-TLV and ER-HOP IPv4 Prefix

RFC 8395, *Extensions to BGP-Signaled Pseudowires to Support Flow-Aware Transport Labels*

14.22 Quality of Service (QoS)

RFC 2430, *A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)*

RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*

RFC 2597, *Assured Forwarding PHB Group*

RFC 3140, *Per Hop Behavior Identification Codes*

RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*

14.23 Remote Authentication Dial In User Service (RADIUS)

draft-oscca-cfrg-sm3-02, *The SM3 Cryptographic Hash Function*

RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*

RFC 2866, *RADIUS Accounting*

RFC 3162, *RADIUS and IPv6*

RFC 6613, *RADIUS over TCP* – with TLS

RFC 6614, *Transport Layer Security (TLS) Encryption for RADIUS*

RFC 6929, *Remote Authentication Dial-In User Service (RADIUS) Protocol Extensions*

14.24 Resource Reservation Protocol - Traffic Engineering (RSVP-TE)

RFC 2702, *Requirements for Traffic Engineering over MPLS*

RFC 2747, *RSVP Cryptographic Authentication*

RFC 2961, *RSVP Refresh Overhead Reduction Extensions*

RFC 3097, *RSVP Cryptographic Authentication -- Updated Message Type Value*

RFC 3209, *RSVP-TE: Extensions to RSVP for LSP Tunnels*

RFC 4124, *Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering*

RFC 4125, *Maximum Allocation Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering*

RFC 4127, *Russian Dolls Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering*

RFC 4561, *Definition of a Record Route Object (RRO) Node-Id Sub-Object*

RFC 5817, *Graceful Shutdown in MPLS and Generalized MPLS Traffic Engineering Networks*

14.25 Routing Information Protocol (RIP)

RFC 1058, *Routing Information Protocol*

RFC 2080, *RIPng for IPv6*

RFC 2082, *RIP-2 MD5 Authentication*

RFC 2453, *RIP Version 2*

14.26 Segment Routing (SR)

RFC 8287, *Label Switched Path (LSP) Ping/Traceroute for Segment Routing (SR) IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs) with MPLS Data Planes*

RFC 8426, *Recommendations for RSVP-TE and Segment Routing (SR) Label Switched Path (LSP) Coexistence*

RFC 8476, *Signaling Maximum SID Depth (MSD) Using OSPF* – node MSD

RFC 8491, *Signaling Maximum SID Depth (MSD) Using IS-IS* – node MSD

RFC 8660, *Segment Routing with the MPLS Data Plane*

RFC 8661, *Segment Routing MPLS Interworking with LDP*

RFC 8665, *OSPF Extensions for Segment Routing*

RFC 8667, *IS-IS Extensions for Segment Routing*

RFC 8669, *Segment Routing Prefix Segment Identifier Extensions for BGP*

RFC 9256, *Segment Routing Policy Architecture*

RFC 9350, *IGP Flexible Algorithm*

14.27 Simple Network Management Protocol (SNMP)

draft-blumenthal-aes-usm-04, *The AES Cipher Algorithm in the SNMP's User-based Security Model* – CFB128-AES-192 and CFB128-AES-256

draft-ietf-isis-wg-mib-06, *Management Information Base for Intermediate System to Intermediate System (IS-IS)*

draft-ietf-mpls-ldp-mib-07, *Definitions of Managed Objects for the Multiprotocol Label Switching, Label Distribution Protocol (LDP)*

draft-ietf-mpls-lsr-mib-06, *Multiprotocol Label Switching (MPLS) Label Switching Router (LSR) Management Information Base Using SMIv2*

draft-ietf-mpls-te-mib-04, *Multiprotocol Label Switching (MPLS) Traffic Engineering Management Information Base*

draft-ietf-ospf-mib-update-08, *OSPF Version 2 Management Information Base*

draft-ietf-vrrp-unified-mib-06, *Definitions of Managed Objects for the VRRP over IPv4 and IPv6* – IPv6

ESO-CONSORTIUM-MIB revision 200406230000Z, *esoConsortiumMIB*

IANA-ADDRESS-FAMILY-NUMBERS-MIB revision 200203140000Z, *ianaAddressFamilyNumbers*

IANAifType-MIB revision 200505270000Z, *ianaifType*

IANA-RTPROTO-MIB revision 200009260000Z, *ianaRtProtoMIB*

IEEE8021-CFM-MIB revision 200706100000Z, *ieee8021CfmMib*

IEEE8021-PAE-MIB revision 200101160000Z, *ieee8021paeMIB*

IEEE8023-LAG-MIB revision 200006270000Z, *lagMIB*

LLDP-MIB revision 200505060000Z, *lldpMIB*

RFC 1157, *A Simple Network Management Protocol (SNMP)*

RFC 1212, *Concise MIB Definitions*

RFC 1215, *A Convention for Defining Traps for use with the SNMP*

RFC 1724, *RIP Version 2 MIB Extension*

RFC 1901, *Introduction to Community-based SNMPv2*

RFC 2021, *Remote Network Monitoring Management Information Base Version 2 using SMIv2*

RFC 2206, *RSVP Management Information Base using SMIv2*

RFC 2578, *Structure of Management Information Version 2 (SMIv2)*

RFC 2579, *Textual Conventions for SMIv2*

RFC 2580, *Conformance Statements for SMIv2*

RFC 2787, *Definitions of Managed Objects for the Virtual Router Redundancy Protocol*

RFC 2819, *Remote Network Monitoring Management Information Base*

RFC 2856, *Textual Conventions for Additional High Capacity Data Types*

RFC 2863, *The Interfaces Group MIB*

RFC 2864, *The Inverted Stack Table Extension to the Interfaces Group MIB*

RFC 2933, *Internet Group Management Protocol MIB*

RFC 3014, *Notification Log MIB*

RFC 3273, *Remote Network Monitoring Management Information Base for High Capacity Networks*

RFC 3410, *Introduction and Applicability Statements for Internet Standard Management Framework*

RFC 3430, *Simple Network Management Protocol (SNMP) over Transmission Control Protocol (TCP) Transport Mapping*

RFC 3411, *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*

RFC 3412, *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)*

RFC 3413, *Simple Network Management Protocol (SNMP) Applications*

RFC 3414, *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*

RFC 3415, *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)*

RFC 3416, *Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)*

RFC 3417, *Transport Mappings for the Simple Network Management Protocol (SNMP)* – SNMP over UDP over IPv4

RFC 3418, *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*

RFC 3419, *Textual Conventions for Transport Addresses*

RFC 3434, *Remote Monitoring MIB Extensions for High Capacity Alarms*

RFC 3584, *Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework*

RFC 3593, *Textual Conventions for MIB Modules Using Performance History Based on 15 Minute Intervals*

RFC 3635, *Definitions of Managed Objects for the Ethernet-like Interface Types*

RFC 3826, *The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model*

RFC 3877, *Alarm Management Information Base (MIB)*

RFC 4001, *Textual Conventions for Internet Network Addresses*

RFC 4022, *Management Information Base for the Transmission Control Protocol (TCP)*

RFC 4113, *Management Information Base for the User Datagram Protocol (UDP)*

RFC 4273, *Definitions of Managed Objects for BGP-4*

RFC 4292, *IP Forwarding Table MIB*

RFC 4293, *Management Information Base for the Internet Protocol (IP)*

RFC 4878, *Definitions and Managed Objects for Operations, Administration, and Maintenance (OAM) Functions on Ethernet-Like Interfaces*

RFC 7420, *Path Computation Element Communication Protocol (PCEP) Management Information Base (MIB) Module*

RFC 7630, *HMAC-SHA-2 Authentication Protocols in the User-based Security Model (USM) for SNMPv3*

14.28 Timing

RFC 3339, *Date and Time on the Internet: Timestamps*

RFC 5905, *Network Time Protocol Version 4: Protocol and Algorithms Specification*

RFC 8573, *Message Authentication Code for the Network Time Protocol*

14.29 Two-Way Active Measurement Protocol (TWAMP)

RFC 5357, *A Two-Way Active Measurement Protocol (TWAMP)* – server, unauthenticated mode

RFC 5938, *Individual Session Control Feature for the Two-Way Active Measurement Protocol (TWAMP)*

RFC 6038, *Two-Way Active Measurement Protocol (TWAMP) Reflect Octets and Symmetrical Size Features*

RFC 8545, *Well-Known Port Assignments for the One-Way Active Measurement Protocol (OWAMP) and the Two-Way Active Measurement Protocol (TWAMP)* – TWAMP

RFC 8762, *Simple Two-Way Active Measurement Protocol* – unauthenticated

RFC 8972, *Simple Two-Way Active Measurement Protocol Optional Extensions* – unauthenticated

RFC 9503, *Simple Two-Way Active Measurement Protocol (STAMP) Extensions for Segment Routing Networks* – excluding Sections 3, 4.1.2 and 4.1.3

14.30 Virtual Private LAN Service (VPLS)

RFC 4761, *Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling*

RFC 4762, *Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling*

RFC 5501, *Requirements for Multicast Support in Virtual Private LAN Services*

RFC 6074, *Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)*

RFC 7117, *Multicast in Virtual Private LAN Service (VPLS)*

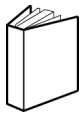
14.31 Yet Another Next Generation (YANG)

RFC 6991, *Common YANG Data Types*

RFC 7950, *The YANG 1.1 Data Modeling Language*

RFC 7951, *JSON Encoding of Data Modeled with YANG*

Customer document and product support



Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback