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

Release 26.10

System Management Guide

3HE 22244 AAAC TQZZA
Edition: 01
November 2026

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

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.

This document includes Nokia proprietary and confidential information, which may not be distributed or disclosed to any third parties without the prior written consent of Nokia.

This document is intended for use by Nokia's customers ("You"/"Your") in connection with a product purchased or licensed from any company within Nokia Group of Companies. Use this document as agreed. You agree to notify Nokia of any errors you may find in this document; however, should you elect to use this document for any purpose(s) for which it is not intended, You understand and warrant that any determinations You may make or actions You may take will be based upon Your independent judgment and analysis of the content of this document.

Nokia reserves the right to make changes to this document without notice. At all times, the controlling version is the one available on Nokia's site.

No part of this document may be modified.

NO WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF AVAILABILITY, ACCURACY, RELIABILITY, TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, IS MADE IN RELATION TO THE CONTENT OF THIS DOCUMENT. IN NO EVENT WILL NOKIA BE LIABLE FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL OR ANY LOSSES, SUCH AS BUT NOT LIMITED TO LOSS OF PROFIT, REVENUE, BUSINESS INTERRUPTION, BUSINESS OPPORTUNITY OR DATA THAT MAY ARISE FROM THE USE OF THIS DOCUMENT OR THE INFORMATION IN IT, EVEN IN THE CASE OF ERRORS IN OR OMISSIONS FROM THIS DOCUMENT OR ITS CONTENT.

Copyright and trademark: Nokia is a registered trademark of Nokia Corporation. Other product names mentioned in this document may be trademarks of their respective owners.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

© 2026 Nokia.

Table of contents

1 About this guide 14
1.1 Precautionary and information messages 14
1.2 Conventions 15

2 What's new 17

3 CLI interface 18
3.1 CLI access and use 18
3.1.1 Accessing the CLI 18
3.1.2 Using the CLI help functions 18
3.1.3 Using the CLI command completion 19
3.1.4 Wildcards and ranges 20
3.1.4.1 Wildcards 20
3.1.4.2 Ranges 24
3.1.4.3 Using wildcards and ranges in show route-table 28
3.1.5 CLI control characters and editing keystrokes 29
3.1.6 Closing the CLI 31
3.2 Configuration modes 31
3.2.1 Configuration candidates 32
3.2.1.1 Exclusive mode 32
3.2.1.2 Shared mode 32
3.2.1.3 Private mode 32
3.2.1.4 Name mode 32
3.2.2 Setting the configuration mode 33
3.2.3 Managing configuration conflicts 34
3.2.4 Committing a configuration in candidate mode 34
3.2.5 Deleting configurations 36
3.2.6 Annotating the configuration 36
3.2.7 Discarding a configuration in candidate mode 38
3.3 Administrative commands 38
3.3.1 Pinging a destination IP address 38
3.3.2 Tracing the path to a destination 39
3.3.3 Using Bash 39
3.3.4 Setting commands to execute periodically 40

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.3.5 Using the diff command....42
3.3.5.1 diff netconf-rpc....43
3.3.6 Using the copy command....45
3.3.7 Using the replace command....48
3.3.8 Displaying configuration details....49
3.3.9 Displaying the command tree hierarchy....51
3.3.10 Displaying the state of the configuration....52
3.3.11 Executing configuration statements from a file....53
3.3.12 Collecting technical support data....53
3.4 CLI output formatting and filtering....54
3.4.1 Using Linux command output modifiers....54
3.4.2 Using the **filter** output modifier....56
3.4.3 Using the **eql** output modifier....58
3.4.4 Using the **jq** output modifier....60
3.4.5 Using the **yq** output modifier....61
3.4.6 Using the **as** output modifier....62
3.4.7 Directing output to a file....64
3.5 CLI environment customization....64
3.5.1 Displaying the environment configuration....65
3.5.2 Managing environment settings....66
3.5.3 Configuring command aliases....67
3.5.4 Configuring command completion....69
3.5.5 Configuring pagination....73
3.5.6 Configuring the bottom toolbar....73
3.5.7 Configuring the command history....74
3.5.8 Configuring **vi** editing mode....74
3.5.9 Configuring the engine type....75
3.5.10 Configuring the network instance....76
3.5.11 Configuring the output format....77
3.5.12 Configuring the prompt....77
3.5.13 Configuring the session idle timeout....78
3.5.14 Configuring the data model....79
3.5.15 Prompt and bottom toolbar keywords....79
3.5.16 Configuring default commit and discard options....83

4 YANG data models....85

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

4.1 SR Linux data models....85
4.2 OpenConfig data models....85
4.2.1 Implementation in SR Linux....85
4.2.2 Interaction with SR Linux data models....86
4.2.2.1 Viewing OpenConfig to SR Linux mapping files....86
4.2.3 Viewing the nosimage OpenConfig capability file....88
4.2.4 Enabling access to OpenConfig data models....90
4.2.5 Specifying the data model mode....91
4.2.5.1 Specifying the data model in the CLI....91
4.2.5.2 Specifying the data model in Linux Bash....91
4.2.5.3 Specifying the data model mode for the gRPC server....92
4.2.6 Verifying the OpenConfig operational state....92

5 YANG-modeled operations....93
5.1 YANG model of an action....93
5.2 Executing an action....94
5.3 Finding an action....94
5.4 Displaying action results....97
5.5 Formatting action output....97

6 RPC overview and supporting interfaces....100
6.1 RPC overview....100
6.1.1 gNMI path convention....101
6.2 Configuring a gRPC or JSON server....101

7 gNMI....102
7.1 Common notification messages....102
7.1.1 Timestamps....102
7.1.2 Path prefix....103
7.1.3 Paths....103
7.1.4 Data node values....104
7.1.4.1 Structured data types....105
7.2 Data model selection using gNMI origin....105
7.2.1 CLI configuration with gNMI origin....106
7.3 gNMI Get RPC....107
7.3.1 GetRequest message....107

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

7.3.2 GetResponse message....107
7.4 gNMI Set RPC....108
7.4.1 SetRequest message....108
7.4.2 SetResponse message....109
7.5 gNMI Subscribe RPC....109
7.5.1 SubscribeRequest message....109
7.5.1.1 SubscriptionList message....110
7.5.2 SubscribeResponse message....111
7.6 gNMI Capabilities RPC....111
7.6.1 CapabilityRequest message....112
7.6.2 CapabilityResponse message....112
7.7 Candidate mode....112
7.8 gNMI extensions....112
7.8.1 Commit Confirmed extension....113
7.8.2 Depth extension....113
7.9 gNMI examples....114
7.9.1 gnmi_get examples....114
7.9.2 gnmi_set examples....116
7.9.3 gnmi_cli examples....118
7.9.4 gnmi_capabilities example....121
7.10 gNMI service configuration....121

8 gNOI....122
8.1 gNOI OS service....122
8.1.1 Install RPC....122
8.1.2 Activate RPC....123
8.1.3 Verify RPC....123
8.2 gNOI BootConfig service....123
8.2.1 GetBootConfig RPC....124
8.2.2 SetBootConfig RPC....124
8.3 gNOI FactoryReset service....124
8.3.1 Start RPC....124
8.3.2 gNOI FactoryReset tools CLI command....125
8.4 gNOI File service....125
8.4.1 Get RPC....125
8.4.2 Put RPC....126

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

8.4.3 Stat RPC ... 126
8.4.4 Remove RPC ... 126
8.5 gNOI System service ... 127
8.5.1 Ping RPC ... 127
8.5.2 Traceroute RPC ... 127
8.5.3 Time RPC ... 128
8.5.4 SwitchControlProcessor RPC ... 128
8.5.5 Reboot RPC ... 128
8.5.6 CancelReboot RPC ... 129
8.5.7 RebootStatus RPC ... 129
8.5.8 KillProcess RPC ... 129
8.6 gNOI Healthz service ... 130
8.6.1 Check RPC ... 131
8.6.2 Get RPC ... 132
8.6.3 List RPC ... 132
8.6.4 Acknowledge RPC ... 133
8.6.5 Artifact RPC ... 133
8.6.6 gNOI Healthz CLI commands ... 133
8.7 gNOI Packet Link Qualification service ... 134
8.7.1 Service Call Flow ... 135
8.7.2 Capabilities RPC ... 136
8.7.3 Create RPC ... 137
8.7.4 Get RPC ... 137
8.7.5 Delete RPC ... 137
8.7.6 List RPC ... 138
8.7.7 Packet Link Qualification CLI commands ... 138
8.7.7.1 Configuring a gNOI packet link qualification profile using the CLI ... 138
8.7.7.2 Running a packet link qualification test using the CLI ... 139
8.8 gNOI BGP service ... 140
8.9 gNOI Containerz service ... 141
8.9.1 Deploy RPC ... 142
8.9.2 ListImage RPC ... 142
8.9.3 RemoveImage RPC ... 142
8.9.4 RemoveContainer RPC ... 142
8.9.5 ListContainer RPC ... 143
8.9.6 StartContainer RPC ... 143

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

8.9.6.1 Custom container location placement ... 144
8.9.7 StopContainer RPC ... 146
8.9.8 UpdateContainer RPC ... 146
8.9.9 Log RPC ... 146
8.9.10 CreateVolume RPC ... 147
8.9.11 RemoveVolume RPC ... 147
8.9.12 ListVolume RPC ... 147
8.9.13 StartPlugin RPC ... 148
8.9.14 StopPlugin RPC ... 148
8.9.15 ListPlugin RPC ... 148
8.9.16 RemovePlugin RPC ... 148
8.10 Container resource management using hierarchical cgroups ... 149
8.10.1 Configuring CPU and memory limits using cgroups ... 149
8.10.2 Verifying the cgroup configuration ... 150
8.11 gRPC YANGAction service for YANG-modeled actions ... 151
8.11.1 YANG action example: BGP neighbor hard reset ... 151
8.12 gNOI configuration ... 152
8.12.1 Configuring gNOI services ... 153

9 gNSI ... 154
9.1 gNSI Authz service ... 154
9.1.1 Authz policy ... 154
9.1.2 Authz authorization logic and interactions ... 156
9.1.3 Default TLS profile ... 157
9.1.4 Supported RPCs ... 157
9.1.5 Rotate RPC ... 157
9.1.6 Probe RPC ... 158
9.1.7 Get RPC ... 158
9.1.8 gNSI Authz tools and state commands ... 158
9.1.8.1 Clearing Authz policy counters using the tools command ... 159
9.1.8.2 Probing Authz policy counters using the tools command ... 159
9.1.8.3 Removing an Authz policy using the tools command ... 159
9.1.8.4 Rotating an Authz policy using the tools command ... 159
9.1.8.5 Displaying the currently installed Authz policy ... 160
9.2 gNSI Certz service ... 161
9.2.1 Rotate RPC ... 162

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

9.2.1.1 Rotate RPC use cases ... 162
9.2.2 CanGenerateCSR RPC ... 164
9.2.3 AddProfile RPC ... 165
9.2.4 DeleteProfile RPC ... 165
9.2.5 GetProfileList RPC ... 165
9.2.6 gNSI Certz tools commands ... 165
9.2.6.1 Rotating a Certz profile using the tools command ... 166
9.2.6.2 Removing a Certz profile using the tools command ... 166
9.2.7 gNSI Certz state commands ... 166
9.3 gNSI Acctz service ... 167
9.3.1 RecordSubscribe RPC ... 168
9.4 gNSI EnrollZ service ... 168
9.4.1 GetIAKCert RPC ... 169
9.4.2 RotateOIakCert RPC ... 169
9.5 gNSI AttestZ service ... 170
9.5.1 AttestRequest RPC ... 170
9.6 gNSI Pathz service ... 171
9.6.1 Authorization policy ... 171
9.6.2 Interactions with role-based access control ... 173
9.6.3 Supported RPCs ... 173
9.6.4 Rotate RPC ... 173
9.6.5 Probe RPC ... 174
9.6.6 Get RPC ... 174
9.7 gNSI configuration ... 174
9.7.1 Configuring gNSI services ... 175

10 JSON interface ... 176
10.1 JSON message structure ... 176
10.1.1 method options ... 177
10.1.2 params options ... 178
10.2 JSON responses ... 179
10.3 Candidate mode ... 180
10.4 Logical expressions ... 180
10.5 Interactive CLI commands ... 180
10.6 JSON examples ... 180
10.6.1 JSON get examples ... 180

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

10.6.2 JSON set examples ... 183
10.6.3 JSON delete example ... 185
10.6.4 JSON validate example ... 186
10.6.5 JSON CLI example ... 187
10.6.6 JSON action examples ... 188

11 NETCONF ... 191
11.1 NETCONF overview ... 191
11.1.1 Transport and sessions ... 191
11.1.2 NETCONF in SR Linux ... 192
11.1.3 YANG-modeled operations - NETCONF ... 192
11.1.3.1 Invoking a YANG action ... 193
11.1.3.2 Synchronous operation support ... 194
11.1.3.3 Streamed action output ... 194
11.1.4 NETCONF IETF monitoring ... 194
11.1.5 YANG library ... 196
11.1.6 Error handling ... 198
11.1.7 Network Management Datastore Architecture (NMDA) ... 199
11.1.8 NETCONF server capabilities ... 200
11.2 NETCONF configuration ... 206
11.2.1 Configuring the SSH transport ... 206
11.2.2 Configuring the NETCONF server ... 206
11.2.2.1 Enabling the NETCONF operations ... 207
11.2.3 Adding filter entries to allow connections to the chosen TCP port ... 208
11.2.4 UNIX socket ... 209
11.2.5 SR Linux logging ... 210
11.2.6 Initiating a connection to a NETCONF server ... 211
11.3 NETCONF state information ... 211
11.4 NETCONF operations ... 212
11.4.1 get ... 213
11.4.2 get-config ... 214
11.4.3 get-data ... 216
11.4.4 get-schema ... 219
11.4.5 edit-config ... 222
11.4.6 edit-data ... 225
11.4.7 copy-config ... 228

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

11.4.8 discard-changes....230
11.4.9 commit....231
11.4.10 cancel-commit....233
11.4.11 lock....234
11.4.12 unlock....235
11.4.13 delete-config....235
11.4.14 close-session....236
11.4.15 validate....237
11.4.16 action....238
11.4.17 kill-session....238
11.5 NETCONF options....239
11.5.1 with-defaults option....239
11.5.2 `with-origin` option....240
11.5.3 Options for the operation attribute....242
11.6 NETCONF filtering....242
11.6.1 Subtree filtering....242
11.6.1.1 Namespace selection and wildcarding....243
11.6.1.2 Containment nodes....246
11.6.1.3 Selection nodes....247
11.6.1.4 Content match nodes....248
11.6.1.5 Subtree filter processing....249

12 SNMP....250
12.1 SNMP architecture....250
12.2 Management information base....250
12.3 SNMP network instance configuration....251
12.4 SNMP versions and configuration....251
12.4.1 SNMPv3 authentication and privacy protocols....251
12.4.2 Configuring SNMPv2c....252
12.4.3 Configuring SNMPv3....253

13 Configuration secrets....256
13.1 Passwords....256
13.2 Shared secrets....257
13.3 Primary secret....257

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

14 Modular Database (MDB)..... 259
14.1 MDB profiles..... 259
14.2 Configuring an MDB profile..... 260

15 General and operational commands..... 261

16 Pre-defined show reports..... 272
16.1 ACL show reports..... 272
16.1.1 ACL descriptions..... 272
16.2 ARP/ND show reports..... 272
16.2.1 ARP/ND descriptions..... 273
16.3 Interface show reports..... 273
16.3.1 Interface descriptions..... 273
16.4 IPv4 tunnel show reports..... 274
16.4.1 IPv4 tunnel descriptions..... 274
16.5 IPv6 tunnel show reports..... 274
16.5.1 IPv6 tunnel descriptions..... 275
16.6 LAG show reports..... 275
16.6.1 LAG descriptions..... 275
16.7 MPLS show reports..... 276
16.7.1 MPLS descriptions..... 276
16.8 Network-instance show reports..... 276
16.8.1 Network-instance descriptions..... 281
16.9 OAM show reports..... 297
16.9.1 OAM descriptions..... 298
16.10 Platform show reports..... 299
16.10.1 Platform descriptions..... 299
16.11 QoS show reports..... 300
16.11.1 QoS descriptions..... 300
16.12 System show reports..... 300
16.12.1 System descriptions..... 301
16.13 Tunnel show reports..... 302
16.13.1 Tunnel descriptions..... 302
16.14 Tunnel-interface show reports..... 303
16.14.1 Tunnel-interface descriptions..... 303

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

16.15 Version show reports....304
16.15.1 Version descriptions....304

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

1 About this guide

This document describes the interfaces used with the Nokia Service Router Linux (SR Linux). These include:

- CLI
- gNMI
- JSON
- NETCONF

This document also provides an overview to CLI plug-ins and details Nokia defined general and operation commands, and show commands.

This document is intended for network technicians, administrators, operators, service providers, and others who need to understand how to interface with the SR Linux.

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

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

1.1 Precautionary and information messages

The following are information symbols used in the documentation.

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

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

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

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

1.2 Conventions

The Nokia SR Linux documentation uses the following command conventions:

- **Bold** type indicates a command that the user must enter.
- Input and output examples are displayed in `Courier` text.
- A vertical bar (|) indicates a mutually exclusive argument.
- Square brackets ([ ]) indicate optional elements.
- Braces ({ }) indicate a required choice. When braces are contained within square brackets, they indicate a required choice within an optional element.
- *Italic* type indicates a variable.

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



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

Table 1: Platform grouping legend

Platform group	Description
7215 IXS	7215 IXS-A1[1]
7220 IXR	All 7220 IXR platforms
7220 IXR-D*x*	7220 IXR-D1, 7220 IXR-D2, 7220 IXR-D2L, 7220 IXR-D3, 7220 IXR-D3L, 7220 IXR-D4, 7220 IXR-D5
7220 IXR-H*x*	7220 IXR-H2, 7220 IXR-H3, 7220 IXR-H4, 7220 IXR-H4-32D, 7220 IXR-H5-32D, 7220 IXR-H5-64D, 7220 IXR-H5-64O, 7220 IXR-H5-64O-512, 7220 IXR-H6-64
7250 IXR[2]	7250 IXR platforms
7250 IXR Gen 2	7250 IXR-6, 7250 IXR-10
7250 IXR Gen 2c+	7250 IXR-6e with IMM2, 7250 IXR-10e with IMM2, 7250 IXR-X1b, 7250 IXR-X3b
7250 IXR Gen 3	7250 IXR-6e with IMM3, 7250 IXR-10e with IMM3, 7250 IXR-18e, 7250 IXR-X4, 7250 IXR-X4 OSFP

1 References to 7215 IXS may be appended with (AA variant) or (AB variant) to indicate the chassis variant. Both variants are 7215 IXS platforms; they are identified by part numbers ending with AA or AB.

2 References to the 7250 IXR platform group may be appended with (including mixed systems) or (excluding mixed systems) to indicate support for a mix of 7250 IXR Gen 2c+ (IMM2) and 7250 IXR Gen 3 (IMM3) line cards in the same chassis.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Platform group	Description
7250 IXR-6e/10e (mixed system)	7250 IXR-6e (mixed system)[3], 7250 IXR-10e (mixed system)[3]

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

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

2 What's new

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

Table 2: What's new in Release 26.10.1

Topic	Location
YANG-modeled operations	YANG-modeled operations
gNOI BootConfig service	gNOI BootConfig service
For dual-CPM chassis, the **Location** field in the StartContainer RPC specifies which CPMs to run the container on	Custom container location placement
JSON-RPC support for YANG modeled actions	JSON interface
NETCONF support of YANG modeled actions	YANG-modeled operations - NETCONF
The gRPC YANGAction service is a complement to gNOI for executing YANG-modeled actions over gRPC	gRPC YANGAction service for YANG-modeled actions
gNOI FactoryReset tools CLI command	gNOI FactoryReset tools CLI command
Encryption of configuration secrets based on primary secret	Configuration secrets
New show reports added for • IPv4 tunnels • IPv6 tunnels • BGP: BMP, malformed updates, FlowSpec v1, origin-validation-state, and RPKI • QoS statistics	See: • IPv4 tunnel show reports • IPv6 tunnel show reports • Network-instance show reports • QoS show reports

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3 CLI interface

The command line interface (CLI) is an interface for configuring, monitoring, and maintaining SR Linux. This chapter describes basic features of the CLI and how to use them.

3.1 CLI access and use

The following sections describe how to access and use the CLI.

3.1.1 Accessing the CLI

About this task

After the SR Linux device is initialized, you can access the CLI using a console or SSH connection. See the SR Linux hardware documentation for information about establishing a console connection and enabling and connecting to an SSH server.

Procedure

Use the following command to connect to the SR Linux and open the CLI using SSH:

ssh admin@*<IP Address>*

Example:

```
$ ssh admin@172.16.0.3
Last login: Fri Sep 20 18:59:50 2024 from 172.16.0.10
Loading environment configuration file(s): []
Welcome to the Nokia SR Linux CLI.

--{ running }--[  ]--
A:admin@srl1#
```

3.1.2 Using the CLI help functions

About this task

The CLI help functions (**?** and **help**) can assist in understanding command usage and indicate which configuration mode you are in.

Procedure

Enter a question mark (**?**) after a command to display the command usage.

Enter **help** at the top level to show the current configuration mode and details on other configuration modes.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: ?

In this example, entering a question mark after the command **network-instance**, shows its usage.

```
# network-instance ?
usage: network-instance <name>
Network instances configured on the local system
Positional arguments:
  name              [string] A unique name identifying the network instance
```

Example: help

The following example shows the system output from the **help** command

```
# help
--------------------------------------------------------------------------------
You are now in the running mode.
Here you can navigate and query the running configuration.
This configuration has been validated, committed and send to the applications.
There are multiple modes you can enter while using the CLI.
Each mode offers its own set of capabilities:
    - running mode: Allows traversing and inspecting of the running configuration.
    - state mode: Allows traversing and inspecting of the state.
    - candidate mode: Allows editing and inspecting of the configuration.
    - show mode: Allows traversing and executing of custom show routines.
To switch mode use 'enter <mode-name>', e.g. 'enter candidate'
To navigate around, you can simply type the node name to enter a context, while
'exit [all]' is used to navigate up.
'{' and '}' are an alternative way to enter and leave contexts.
'?' can be used to see all possible commands, or alternatively,
'<TAB>' can be used to trigger auto-completion.
'tree' displays the tree of possible nodes you can enter.
--------------------------------------------------------------------------------
--{ * running }--[  ]--
```

Related topics

Configuration modes

3.1.3 Using the CLI command completion

About this task

To reduce keystrokes or aid in remembering a command name, use the CLI command completion.

Procedure

Press the **Tab** key at any mode or level to complete the next command level.

When a command is partially entered, the remainder of the command appears ahead of the prompt in lighter text. Press the **Tab** key to complete the command.



sc0171

When the **Tab** key is pressed and there are multiple options, the options are shown in a pop-up:

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
# info
       ..                depth             from              system
       /                 detail            interface         use-proto-json
       debug             flat              network-instance
```

sc0170

Related topics

Configuring command completion

3.1.4 Wildcards and ranges

Rather than define a single value for a parameter, the SR Linux CLI allows you to enter a wildcard or a range to substitute for one or more values, such as a list of interfaces. The CLI engine automatically expands the specified wildcard or range into a list of individual values and executes the command for each value in that list.

3.1.4.1 Wildcards

In the SR Linux CLI, you can enter an asterisk (*) to serve as a wildcard character in commands. You can use wildcards to represent any predefined values for a parameter. Note the following considerations when you use wildcards:

- Wildcards are valid in both **info** and configuration commands.
- Multiple wildcards are allowed in one command.
- Wildcards apply only to existing pre-configured objects (for non-configured objects, use ranges).
- Wildcards expand to YANG list keys, which must be valid in the context in which the wildcard is entered.
- Wildcards and ranges can be entered in the same command.

The following sections provide some examples of wildcards.

Wildcards in info commands

The following example uses a wildcard character as a substitute for the subinterface value in the **info interface** command to display all subinterfaces of ethernet-1/1:

Example: Display all subinterfaces using a wildcard

```
--{ * candidate shared default }--[  ]--
# info with-context interface ethernet-1/1 subinterface *
    interface ethernet-1/1 {
        subinterface 0 {
            admin-state enable
        }
        subinterface 1 {
            admin-state enable
        }
        subinterface 2 {
            admin-state enable
        }
    }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

You can also enter multiple wildcards to substitute for multiple parameters. The following example checks the active status for all IPv4 unicast BGP routes in the default network instance:

Example: Display state for all IPv4 unicast BGP routes using wildcards

```
--{ * candidate shared default }--[  ]--
# info from state with-context network-instance default route-table ipv4-unicast route * id * route-
type bgp route-owner * origin-network-instance * active
    network-instance default {
        route-table {
            ipv4-unicast {
                route 10.0.0.2/32 id 0 route-type bgp route-owner bgp_mgr origin-network-instance
 default {
                    active true
                }
                route 10.0.0.3/32 id 0 route-type bgp route-owner bgp_mgr origin-network-instance
 default {
                    active true
                }
                route 10.0.0.4/32 id 0 route-type bgp route-owner bgp_mgr origin-network-instance
 default {
                    active true
                }
            }
        }
    }
```

Wildcards in configuration commands

You can also use wildcards in configuration commands. The following example adds **subinterface 0** to every configured interface in the system:

Example: Add subinterface 0 to all configured interfaces

```
--{ + candidate shared default }--[  ]--
# /interface * subinterface 0 description "created with a wildcard"
```

Wildcards in interface commands

Wildcards can expand YANG list keys, with the exception of interface names. However, within the interface name, you can use wildcards for either the linecard or port elements.

The following example uses a wildcard for the port value to display the admin-state for all interfaces on linecard 1:

Example: Display admin-state for all interfaces on linecard 1

```
--{ + candidate shared default }--[  ]--
# info with-context interface ethernet-1/* admin-state
    interface ethernet-1/1 {
        admin-state enable
    }
    interface ethernet-1/3 {
        admin-state enable
    }
    interface ethernet-1/8 {
        admin-state enable
    }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Similarly, you can include a wildcard to specify one port on all linecards (for example: **interface ethernet-*/2**).

Wildcards in expanded contexts

Wildcards support the ability to enter an expanded context. Context-based configuration workflows with wildcards can eliminate copying and pasting by applying the configuration commands to all existing objects.

For example, to analyze the state of the configured interfaces, you can enter the context of all interfaces at once, as shown in the following example:

Example: Enter state mode for all interfaces

```
--{ + running }--[  ]--
# enter state
--{ + state }--[  ]--
# interface *
--{ + state }--[ interface * ]--
```

From the wildcarded context, you have access to a range of commands that are executed across all applicable interfaces. The following example lists the number of incoming unicast packets on all interfaces:

Example: List the number of incoming unicast packets on all interfaces

```
--{ + state }--[ interface * ]--
# info statistics in-unicast-packets
    interface ethernet-1/1 {
        statistics {
            in-unicast-packets 0
        }
    }
    interface ethernet-1/3 {
        statistics {
            in-unicast-packets 0
        }
    }
    ...
    interface mgmt0 {
        statistics {
            in-unicast-packets 919
        }
    }
```

You can also use wildcards in the context mode for configuration tasks. The following example adds VLAN tagging for **subinterface 0** on all applicable interfaces:

Example: Add VLAN tagging for subinterface 0 on all interfaces

```
# switching to candidate datastore
--{ + state }--[  ]--
A:srl# enter candidate

# entering the wildcarded context
# of subinterface 0 of all interfaces
--{ + candidate shared default }--[  ]--
A:srl# interface * subinterface 0

# adding vlan tagging on all of them
--{ + candidate shared default }--[ interface * subinterface 0 ]--
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
A:srl# vlan encap single-tagged vlan-id optional
```

As a result, the VLAN tagging configuration is applied to all subinterfaces:

Example: Confirm the VLAN tagging is applied (diff)

```
--{ +* candidate shared default }--[ interface * subinterface 0 ]--
# diff
      interface ethernet-1/1 {
          subinterface 0 {
              vlan {
                  encap {
+                     single-tagged {
+                         vlan-id optional
+                     }
                  }
              }
          }
      }
      interface ethernet-1/3 {
          subinterface 0 {
              vlan {
                  encap {
+                     single-tagged {
+                         vlan-id optional
+                     }
                  }
              }
          }
      }
...
```

Wildcards and strings

Wildcards can also be used with string-based keys. You can add a wildcard character anywhere in the string key to match any number of characters in that position. For example, on a system that has several VRFs with names beginning with `red`, you can match multiple VRFs as in the following example:

Example: Match multiple VRFs

```
--{ +* candidate shared default }--[  ]--
A:srl# info network-instance red*
    network-instance red {
        admin-state enable
    }
    network-instance red-a {
        admin-state enable
    }
    network-instance red-b {
        admin-state enable
    }
    network-instance red1 {
        admin-state enable
    }
```

In this case, **red*** expands to the **red**, **red-a**, **red-b**, and **red1** keys.

Wildcard expansion with string keys also provides a means to filter out objects that match a particular pattern, such as a customer name or location code.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Wildcard applicability and scope

It is important to note that wildcards expand to existing objects only. For example, if your candidate or running datastore has only two interfaces **ethernet-1/1** and **ethernet-1/5**, then the wildcard **ethernet-1/*** only matches these existing interfaces.

Also, to expand a list key, the list key must pertain to the context in which the wildcard is employed.

Consider the case where three interfaces are configured and you want to enable LLDP. First, you can ensure that the interfaces exist:

Example: List interface state

```
--{ + candidate shared default }--[  ]--
# info from running interface ethernet-1/* admin-state
    interface ethernet-1/1 {
        admin-state enable
    }
    interface ethernet-1/3 {
        admin-state enable
    }
    interface ethernet-1/8 {
        admin-state enable
    }
```

But if you try to enable LLDP on all interfaces using a wildcard, the operation fails:

Example: Attempt to enable LLDP using a wildcard

```
--{ + candidate shared default }--[  ]--
# system lldp interface ethernet-1/* admin-state enable
Error: Path '.system.lldp.interface{.name==ethernet-1/*}' does not specify any existing
 objects
```

This error occurs because the **/system/lldp/interface** list does not contain these interfaces within its own context. These interfaces are instead contained in the **/interface** list context, which the **/system/lldp/interface** list references. As a result of this referencing structure, these interfaces are not eligible for wildcard expansion in the **/system/lldp/interface** context.

In this case, ranges can be useful as they can create objects that do not exist yet.

3.1.4.2 Ranges

CLI ranges allow you to define a series of values for a particular list key. Unlike wildcards, ranges can generate new objects within the system, allowing for bulk object creation.

To express a range, enter a list of values separated by commas. Each value can be a single scalar value. You can also specify a continuous range of values using the double-dot (**..**) delimiter, or mix both formats within the same range specification.

Note the following considerations when you use ranges:

- Ranges are valid in both **info** and configuration commands.
- Multiple ranges are allowed in one command.
- Ranges apply to both existing and new objects.
- In a configuration command, if a range includes an object that already exists, the command overwrites the existing object.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- Wildcards and ranges can be entered in the same command.

The following table provides a few examples showing the syntax of different range patterns and how they translate to the list of elements:

Syntax	Result
{1,3}	1, 3
{2..5}	2, 3, 4, 5
{1,3..5,8}	1, 3, 4, 5,8

The following example shows how to display the administrative state of interfaces 1, 3, and 5 using a comma-separated list of elements in just one command:

Example: Display admin-state of interfaces 1,3, and 5

```
--{ + running }--[  ]--
# info interface ethernet-1/{1,3,5} admin-state
    interface ethernet-1/1 {
        admin-state enable
    }
    interface ethernet-1/3 {
        admin-state enable
    }
    interface ethernet-1/5 {
        admin-state enable
    }
```

To create a consecutive range of integer values, you can use double-dot (**..**) notation. The following example uses a range of **ethernet-1/{2..4}** to list all interfaces in the range between 2 and 4 (the range boundaries are included).

Example: Display consecutive range of interfaces

```
--{ + running }--[  ]--
# info interface ethernet-1/{2..4} admin-state
    interface ethernet-1/2 {
        admin-state enable
    }
    interface ethernet-1/3 {
        admin-state enable
    }
    interface ethernet-1/4 {
        admin-state enable
    }
```

The following example mixes the two range patterns, providing greater flexibility:

Example: Mix ranges and scalars

```
--{ + running }--[  ]--
# info interface ethernet-1/{1,3..5,8} admin-state
    interface ethernet-1/1 {
        admin-state enable
    }
    interface ethernet-1/2 {
        admin-state enable
    }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
    interface ethernet-1/3 {
        admin-state enable
    }
    interface ethernet-1/4 {
        admin-state enable
    }
    interface ethernet-1/5 {
        admin-state enable
    }
    interface ethernet-1/8 {
        admin-state enable
    }
```

Objects and scoping

Ranges can generate new objects within the system, which is useful for bulk object creation. However, if the CLI range includes an object that already exists, the configuration command overwrites the existing object as well.

Conversely, in the case of an **info** command, if a range expands to a non-existing object, it is skipped.

To demonstrate this behavior, consider a freshly deployed system, where no **ethernet-1/*** interfaces are configured:

Example: Confirm no interfaces configured

```
--{ + running }--[  ]--
# info interface ethernet-1/*
--{ + running }--[  ]--
#
```

To create a set of interfaces, wildcards are not helpful because they cannot expand to undefined objects. Instead, you can define a range:

Example: Create new objects with ranges

```
--{ + running }--[  ]--
# enter candidate

--{ + candidate shared default }--[  ]--
# interface ethernet-1/{1..4} admin-state enable
```

By using ranges in candidate mode, four new interfaces are created on the system with a single command:

Example: Display configuration changes (diff)

```
--{ +* candidate shared default }--[  ]--
# diff
+     interface ethernet-1/1 {
+         admin-state enable
+     }
+     interface ethernet-1/2 {
+         admin-state enable
+     }
+     interface ethernet-1/3 {
+         admin-state enable
+     }
+     interface ethernet-1/4 {
+         admin-state enable
+     }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Because ranges can create new list elements, you can also successfully enable LLDP on multiple interfaces (unlike the LLDP example that failed with wildcards).

Example: Enable LLDP on multiple interfaces

```
--{ +* candidate shared default }--[  ]--
# system lldp interface ethernet-1/{1,2} admin-state enable
```

String key ranges

All the preceding range examples use integer keys, such as interface numbers or VLAN IDs. But ranges also support string-based keys, such as creating multiple named VRFs or ACLs.

Syntax	Result
{red,blue}	"red", "blue"
{red{1..2}}	"red1", "red2"
{red-{a..c}}	"red-a", "red-b", "red-c"

In its simplest form, string-based ranges operate on comma-separated list of strings. In the following example, two strings **red** and **blue** are included in the command to create two network instances.

Example: Create network instances red and blue

```
--{ +* candidate shared default }--[  ]--
A:srl# network-instance {red,blue} admin-state enable
```

The defined range expands to two elements and as a result two VRFs are created:

Example: Display configuration changes (diff)

```
--{ +* candidate shared default }--[  ]--
# diff
+     network-instance blue {
+         admin-state enable
+     }
+     network-instance red {
+         admin-state enable
+     }
```

Advanced templating syntax that involves nested ranges and a combination of both integer and string values is also supported:

Example: Create multiple red and blue VRFs

```
--{ +* candidate shared default }--[  ]--
# network-instance {red{1..2},blue{1..2}} admin-state enable
```

In this case, the defined range expands to two red and two blue VRFs:

Example: Display configuration changes (diff)

```
--{ +* candidate shared default }--[ network-instance {red{1..2},blue{1..2}} ]--
# diff
+     network-instance red1 {
+         admin-state enable
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
+     }
+     network-instance red2 {
+         admin-state enable
+     }
+     network-instance blue1 {
+         admin-state enable
+     }
+     network-instance blue2 {
+         admin-state enable
+     }
```

String-based ranges can create multiple named objects at once, and with nesting, you can construct even more intricate structures.

The **info** command can also take advantage of string-based ranges. The following example displays all ACLs that adhere to a specific naming pattern:

Example: Display ACL filters cust1-* and cust2-*

```
--{ * candidate shared default }--[  ]--
# info acl acl-filter {cust1-*,cust2-*} type ipv4 description
    acl {
        acl-filter cust1-filter1 type ipv4 {
            description somefilter
        }
        acl-filter cust1-filter2 type ipv4 {
            description somefilter
        }
        acl-filter cust2-filter1 type ipv4 {
            description somefilter
        }
        acl-filter cust2-filter2 type ipv4 {
            description somefilter
        }
    }
```

String-based ranges can also be useful in large scale deployments where for example named objects can encode customer or facility information.

3.1.4.3 Using wildcards and ranges in show route-table

Procedure

You can use ranges and wildcards in the **show network-instance route-table** command to display routes for a subset of prefixes or prefix lengths in one or more network instances.

For example, any of the following expressions can match route 192.168.0.1/32:

Table 3: Wildcard and range expressions to match 192.168.0.1/32

Expression	Description
*****	Full wildcard
***.168.0.1/32**	Wildcard for first octet
.168.0.1/	Wildcards for first octet and prefix length
1*.168.0.1/32	Partial wildcard in first octet

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Expression	Description
{190..192}.*	Range for first octet and wildcard for final octets and prefix length

The following example displays routes for 172.18.0.x prefixes that have a prefix length of /24 (**172.18.0.*/24**) and for any prefixes that have a prefix length of /32 (***32**) in either the management (**m***) or default (**def***) network instances.

Example: Filtering route table output

```
--{ candidate shared default }--[  ]--
A:dut1# show network-instance {m*,def*} ipv4 route {172.18.0.*/24,*32}
=========================================================================================================
========
IPv4-unicast route table for ip-vrf network-instance: mgmt
---------------------------------------------------------------------------------------------------------
--------
Flags: > (best), * (unviable), ! (failed)
     : L (leaked route from another network-instance)
     : B (backup NHG active and displayed)
     : S (statistics supported)
     : D (dynamic LB), R (resilient LB)
---------------------------------------------------------------------------------------------------------
--------
Prefix              Route Type   Metric   Pref    Flags    Next-Hop(s)
---------------------------------------------------------------------------------------------------------
--------
172.18.0.0/24       local        0        0       >        172.18.0.3(mgmt0.0)
192.0.2.32/32       static       1        5       >        discard
=========================================================================================================
========
IPv4-unicast route table for default network-instance
---------------------------------------------------------------------------------------------------------
--------
Flags: > (best), * (unviable), ! (failed)
     : L (leaked route from another network-instance)
     : B (backup NHG active and displayed)
     : S (statistics supported)
     : D (dynamic LB), R (resilient LB)
---------------------------------------------------------------------------------------------------------
--------
Prefix              Route Type   Metric   Pref    Flags    Next-Hop(s)
---------------------------------------------------------------------------------------------------------
--------
2.2.2.2/32          isis         10       160     >        1.1.2.2(ethernet-1/1.1)
4.4.4.4/32          isis         20       160     >        1.1.2.2(ethernet-1/1.1)
10.0.0.4/32         bgp          1000     170     >        200.0.0.1(tunnel:bgp)
200.0.0.1/32        bgp-label    0        170     >        mpls(1.1.2.2, ethernet-1/1.1,
 label:250002)
---------------------------------------------------------------------------------------------------------
```

3.1.5 CLI control characters and editing keystrokes

Use control characters and editing keystrokes to move the cursor on the command line to edit, complete, and recall commands previously entered. The following table lists keystrokes to execute and edit commands.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Table 4: CLI keystrokes

Task	Keystroke
Move cursor to the beginning of the line	**Ctrl-A** or **Home**
Move cursor to the end of the line	**Ctrl-E** or **End**
Move cursor one character to the left	**Ctrl-B** or **Left arrow**
Move cursor one character to the right	**Ctrl-F** or **Right arrow**
Move cursor back one word or to the beginning of the current word if cursor is not at the start of the word	**Alt-B** or **Esc-B**
Move cursor back one word	**Alt-D** or **Esc-D**
Move cursor forward one word	**Alt-F** or **Esc-F**
Delete the current character	**Ctrl-D**
Delete the word up to the cursor	**Ctrl-W**
Delete the current character and move the cursor left	**Ctrl-H**
Delete text up to the cursor and preserve the character under the cursor	**Ctrl-U**
Delete the text after the cursor, without preserving the character under the cursor	**Ctrl-K**
Transpose the character to the left of the cursor with the character the cursor is placed on	**Ctrl-T**
Stop the current command	**Ctrl-C**
Enter a tab	**Ctrl-I** or **Tab**
Enter a new line	**Ctrl-J** or **Enter**
Enter a carriage return	**Ctrl-M**
Clear the screen	**Ctrl-L**
Complete a partial command	Enter the first few letters, then press one of the following (when enabled in the environment settings): • **Alt+/** • **Enter** • **Right arrow** • **Spacebar**

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Task	Keystroke
	• **Tab**
Display the previous entry in the command history	**Ctrl-P** or **Up arrow** or **Page up**
Display the next entry in the command history	**Ctrl-N** or **Down** or **Page down**
Search the command history in reverse order	**Ctrl-R**
Display the last word of the previous command	**Esc-.**

Navigate one level up within a context. For example:
`--{running}--[interface ethernet-1/1 subinterface 1]--`
`# exit`
`--{running}--[interface ethernet-1/1]--`
`#`
	exit

Return to the root context. For example:
`--{running}--[interface ethernet-1/1 subinterface 1]--`
`# exit all`
`--{running}--[ ]--`
`#`
	exit all or **/**

3.1.6 Closing the CLI

Procedure

Close the CLI using one of the following methods:

- Press **Ctrl+D**.
- Enter the **quit** command at the CLI prompt.

3.2 Configuration modes

Configuration modes define how the system is running when transactions are performed. Supported modes are the following:

- **Candidate** – Use this mode to modify a configuration. Modifications are not applied to the running system until a **commit** command is issued. When committed, the changes are copied to the running configuration and become active.

  There are different types of configuration candidates. See Configuration candidates.
- **Running** – Use this mode to display the currently running or active configuration. Configurations cannot be edited in this mode.
- **State** – Use this mode to display the configuration and operational states. The state mode displays more information than show mode.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- **Show** – Use this mode to display configured features and operational states. The show mode displays less information than state mode.

3.2.1 Configuration candidates

You can modify the candidate configuration in different modes:

- Exclusive
- Shared
- Private
- Name

3.2.1.1 Exclusive mode

When entering candidate mode, if you specify the **exclusive** keyword, it locks out other users from making changes to the candidate configuration.

You can enter candidate exclusive mode only under the following conditions:

- The current shared candidate configuration has not been modified.
- There are no other users in candidate shared mode.
- No other users have entered candidate exclusive mode.

3.2.1.2 Shared mode

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

A default candidate is defined per user (see Name mode).

Use caution when allowing multiple users access to the candidate configuration at the same time. If one user commits the configuration, that commits the changes made by any other users who have modified the current candidate configuration.

3.2.1.3 Private mode

A private candidate allows multiple users to modify a configuration; however when a user commits their changes, only the changes from that user are committed. When a private candidate is created, private datastores are created and a snapshot is taken from the running database to create a baseline.

When starting a private candidate, a default candidate is defined per user with the name 'private-<username>' unless a unique name is defined (see Name mode).

3.2.1.4 Name mode

Candidate types support an optional name. This allows multiple users to share environments.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

When a candidate is created with a name specified, a new entry is created in the /system/configuration/candidate[] list. Other users can enter the same candidate (if the candidate type is shared) using this name.

If no name is specified, then the name **default** is used. This means the global shared candidate can be accessed by entering **enter candidate name default** or just **enter candidate**, For private candidates the name **default** is not used (because private candidates share a list with shared candidates). Instead, private candidates are created with the name **private-**<*username*>.

Note: A candidate named **default** cannot be created or configured in private mode.

Named candidates are automatically deleted when there are no active sessions present and they are empty, or after 7 days of no session activity. This 7-day default is configurable in the **/system/configuration/idle-timeout** field.

3.2.2 Setting the configuration mode

Procedure

After logging in to the CLI, you are initially placed in running mode. To change between modes, use one of the commands in the following table.

Table 5: Commands to change configuration mode

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

Example: Change from running to shared mode

To change from running to a shared candidate mode (using the default)':

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


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

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

Example: Switch between shared and candidate exclusive modes

To switch between candidate shared and candidate exclusive modes, you must first switch to a different configuration mode (for example, running mode) before entering candidate shared or exclusive mode. For example:

```
--{ running }--[  ]--
# enter candidate exclusive
--{ candidate exclusive }--[  ]--
Are you sure? (y/[n]):
# enter running
--{ running }--[  ]--
# enter candidate
--{ candidate shared default}--[  ]--
```

Example: Enter candidate mode for named candidate

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

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

3.2.3 Managing configuration conflicts

About this task

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

Procedure

You can use the **baseline** command to assist in managing conflicts. It uses the following arguments:

- **baseline update** - Performs an update of the complete baseline datastore, pulling in any changes that occurred in the running datastore since the baseline snapshot was taken.
- **baseline diff**- Shows baseline configuration changes with options to refine by area.
- **baseline check** - Performs a dry-run baseline check, and if conflicts are detected, an informational or warning message is generated.

3.2.4 Committing a configuration in candidate mode

About this task

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

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Procedure

Step 1. Enter candidate mode:

```
# enter candidate
```

Step 2. Enter configuration commands.

Step 3. Enter the **commit** command when with the required option.

Table 6: commit command options

Option	Action	Permitted additional arguments
commit now	Apply the changes, exit candidate mode, and enter running mode.	NA
commit stay	Apply the changes and then remain in candidate mode.	**commit stay [save] [comment] [confirmed]**
commit save	Apply the changes and automatically save the commit to the startup configuration. Can be used other arguments except **now** (for example, **commit stay save**).	**commit [stay] [checkpoint] save [confirmed] [comment]**
commit checkpoint	Apply the changes and cause an automatic checkpoint after the commit succeeds.	**commit [stay] [now] checkpoint [save] [confirmed]**
commit validate	Verify that a propose configuration change passes a management server validation.	NA
commit comment <*comment*>	Use with other keywords (except **validate**) to add a user comment (for example, **commit stay comment** <*comment*> where <*comment*> is a quoted string, 1-255 characters.	**commit [stay] [save] [checkpoint] [confirmed] comment**
commit confirmed **commit confirmed [timeout]** **commit confirmed [accept | reject]**	Apply the changes, but requires an explicit confirmation to become permanent. If the explicit confirmation is not issued within a specified time period, all changes are automatically reverted. The timeout period default is 600 seconds (10 mins.), or can be provisioned with a value of 1-86400 sec.). The timeout parameter cannot be used with the accept or reject parameter. Before the timer expires, the accept parameter explicitly confirms and applies the changes. With no timer running, the reject parameter explicitly rejects the changes.	**commit [checkpoint] [save] [stay] [comment] confirmed**

Example: commit stay and commit confirmed

This example shows the **commit stay** option:

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


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

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

# commit stay
All changes have been committed. Starting new transaction.

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

This example shows the **commit confirmed** option with a custom timeout followed by an accept action.

```
--{ * candidate shared default}--[  ]--
# commit confirmed timeout 86400
Commit confirmed (automatic rollback in a day)
All changes have been committed. Leaving candidate mode.
--{ running }--[  ]--

# commit confirmed accept
Info: Commit confirmed, automatic rollback cancelled
--{ candidate shared default}--[  ]--
#
```

3.2.5 Deleting configurations

Procedure

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

Example: Delete configuration

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

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

3.2.6 Annotating the configuration

Procedure

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

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

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: Add a comment

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

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

To replace the existing comment, use **!!!** instead of **!!** in the command. The **>** character is not supported in comments with these commands.

Example: Add a comment in annotate mode

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

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

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

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

```
--{ running }--[  ]--
# info with-context acl acl-filter ip-tcp type ipv4
    acl {
        acl-filter ip-tcp type ipv4 {
            !!! Filter TCP traffic
            entry 100 {
                action {
                    log true
                    drop {
                    }
                }
            }
            entry 110 {
                action {
                    log true
                    accept {
                    }
                }
            }
        }
    }
```

The comment can also be displayed using the **info flat** command. For example:

```
--{ * candidate shared default }--[  ]--
# info flat with-context acl acl-filter ip-tcp type ipv4
set / acl acl-filter ip-tcp type ipv4 !!! Filter TCP traffic
set / acl acl-filter ip-tcp type ipv4 entry 100 action log true
set / acl acl-filter ip-tcp type ipv4 entry 100 action drop
set / acl acl-filter ip-tcp type ipv4 entry 110 action log true
set / acl acl-filter ip-tcp type ipv4 entry 110 action accept
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

In **diff** mode, the comment is displayed as follows:

```
--{ +* candidate shared default }--[ acl acl-filter ip-tcp type ipv4 ]--
# diff
+     !!! Filter TCP traffic
```

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

3.2.7 Discarding a configuration in candidate mode

Procedure

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

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

Example: Discard changes and remain in candidate mode

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

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

3.3 Administrative commands

The common administrative CLI commands in this section can help you understand a current configuration and perform routine configuration tasks.

3.3.1 Pinging a destination IP address

Procedure

Use the **ping** (IPv4) or **ping6** (IPv6) command to contact an IP address. Use this command in any mode.

Example: ping for IPV4

```
--{ running }--[  ]--
# ping 192.168.1.1 network-instance default
Pinging 192.168.1.1 in srbase-default
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.027 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.030 ms
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 6165ms
rtt min/avg/max/mdev = 0.027/0.030/0.033/0.005 ms
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.3.2 Tracing the path to a destination

Procedure

To display the path a packet takes to a destination, use the **traceroute** (IPv4) or **traceroute6** (IPv6) command.

To trace the route using TCP SYN packets instead of UDP or ICMP echo packets, use the **tcptraceroute** command.

Example: traceroute for IPv4

```
--{ running }--[  ]--
# traceroute 1.1.1.1  network-instance mgmt
Using network instance srbase-mgmt
traceroute to 1.1.1.1 (1.1.1.1), 30 hops max, 60 byte packets
 1  172.18.18.1 (172.18.18.1)  1.268 ms  1.260 ms  1.256 ms
 2  172.21.40.1 (172.21.40.1)  1.253 ms  1.848 ms  1.851 ms
 3  172.22.35.230 (172.22.35.230)  1.835 ms  1.834 ms  1.828 ms
 4  66.201.62.1 (66.201.62.1)  3.222 ms  3.222 ms  3.216 ms
 5  66.201.34.17 (66.201.34.17)  5.474 ms  5.475 ms  5.480 ms
 6  * * *
 7  206.81.81.10 (206.81.81.10)  32.577 ms  32.542 ms  32.400 ms
 8  1.1.1.1 (1.1.1.1)  22.627 ms  22.637 ms  22.638 ms
```

3.3.3 Using Bash

Procedure

Use the **bash** command to execute a Linux Bash (Bourne-Again SHell) session. To exit Bash and return to the CLI, enter **exit**.

You can also use the **bash** command to enter Linux commands directly from the SR Linux CLI prompt.



Note:
Use the Linux **sudo** command to execute commands with root privileges when **system aaa authentication user** *username* **superuser** is enabled in the local user's profile.

Example: Using Bash

```
--{ running }--[  ]--
# bash
[root@3-node_srlinux-A /]# ls
anaconda-post.log  dev            etc   lib    media  opt   root  sbin  sys   tmp  var
bin                entrypoint.sh  home  lib64  mnt    proc  run   srv   tini  usr
[root@3-node_srlinux-A /]# exit
logout
--{ running }--[  ]--
#
```

Example: Enter Linux commands using Bash

```
--{ running }--[  ]--
# bash cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.18.18.5     3-node-srlinux-A
### SRLINUX - ANYTHING MODIFIED BEYOND THIS POINT WILL BE OVERWRITTEN ###
127.0.0.1 3-node-srlinux-A
### SRLINUX FOOTER ###
```

3.3.4 Setting commands to execute periodically

Procedure

To set a command to execute periodically, use the **watch** command. The **watch** command can be used with any valid command with the exception of interactive commands (for example, **ping**, **monitor**, **packet-trace**, **bash**, and bash oneliners such as **bash cat /etc/hosts**). An error message is generated if an interactive command is used. Output redirection is supported.

When used, the first page of output displays. The command then re-executes every 2 seconds by default.

The watch command uses the following format:

watch <*arguments*> <*command to watch*>

All arguments must precede the command being watched. Arguments for the **watch** command are:

Arguments	Definition
interval	Sets an interval for the command to re-execute. Range: 1 - 3600 seconds. Default: 2 seconds
differences	For output that is actively changing, highlights the differences between the previous and current execution
cumulative-differences	For output that is actively changing, highlights the differences between the first execution and the current execution
no-colors	When used with differences or cumulative-differences, markers are used to show differences (verses colors)
no-limit	Allows the full output of each execution to display (verses limiting to the height of the terminal)
no-paging	Provides continuous output (verses redrawing the terminal with each execution)
no-title	Turns off the header that shows the interval, command, number of executions that have occurred, and the time

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

The command being watched does not require double quotes. In addition, the auto complete function can be used to complete a valid command.

To exit the **watch** command, enter **Ctrl-C**.

Example: Watch interface statistics

In the following example, the watch command is used to show interface statistics (showing in-octets and out-octets in table format). The interval is modified to 10 seconds.

```
# watch interval 10 info from state interface * statistics | as table |filter fields in-
octets out-octets

Every 10.0s: info from state interface * statistics | as table | filter fields in-octets
 out-octets (Executions 2,
Thu 04:47:34PM)
+--------------------+----------------------+----------------------+
|      Interface     |      In-octets       |      Out-octets      |
+====================+======================+======================+
| ethernet-1/1       |               812755 |               812626 |
| ethernet-1/2       |               811411 |               810362 |
| ethernet-1/3       |                      |                      |
| ethernet-1/4       |                      |                      |
| ethernet-1/5       |                      |                      |
| ethernet-1/6       |                      |                      |
| ethernet-1/7       |                      |                      |
| ethernet-1/8       |                      |                      |
| ethernet-1/9       |                      |                      |
| ethernet-1/10      |                      |                      |
| ethernet-1/11      |                      |                      |
| ethernet-1/12      |                      |                      |
| ethernet-1/13      |                      |                      |
| ethernet-1/14      |                      |                      |
| ethernet-1/15      |                      |                      |
```

Example: Show differences in interface statistics

In the following example, the watch command is used to show differences in interface statistics (with in-octets and out-octets in table format). In addition, the no-color argument is used to show the differences between previous and current execution with markers verses color indicators.

```
# watch differences no-colors info from state interface * statistics | as table |filter
 fields in-octets out-octets

Every 10.0s: info from state interface * statistics | as table | filter fields in-octets
 out-octets (Executions 4,
 Thu 04:54:48PM)
  +--------------------+----------------------+----------------------+
  |      Interface     |      In-octets       |      Out-octets      |
  +====================+======================+======================+
- | ethernet-1/1       |               817889 |               817760 |
?                                       --                      ^ ^
+ | ethernet-1/1       |               817975 |               817865 |
?                                        ++                     ^ ^
- | ethernet-1/2       |               816328 |               815088 |
?                                       ^^                      ^^
+ | ethernet-1/2       |               816585 |               815278 |
?                                       ^ +                     ^^
  | ethernet-1/3       |                      |                      |
  | ethernet-1/4       |                      |                      |
  | ethernet-1/5       |                      |                      |
  | ethernet-1/6       |                      |                      |
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
  | ethernet-1/7        |                     |                     |
  | ethernet-1/8        |                     |                     |
  | ethernet-1/9        |                     |                     |
```

3.3.5 Using the diff command

Procedure

Use the **diff** command to get a comparison of configuration changes. Optional arguments can be used to indicate the source and destination datastore. The following use rules apply:

- If no arguments are specified, the diff is performed from the candidate to the baseline of the candidate.
- If a single argument is specified, the diff is performed from the current candidate to the specified candidate.
- If two arguments are specified, the first is treated as the source, and the second as the destination.

Global arguments include: baseline, candidate, checkpoint, factory, file, from, rescue, running, and startup.

The **diff** command can be used outside of candidate mode, but only if used with arguments.

Example: Basic diff command with no arguments

The following shows a basic **diff** command without arguments. In this example, the description and admin state of an interface are changed and the differences shown:

```
--{ candidate shared default }--[  ]--
# interface ethernet-1/1 admin-state disable
--{ * candidate shared default }--[  ]--
# interface ethernet-1/2 description "updated"
--{ * candidate shared default }--[  ]--
# diff
      interface ethernet-1/1 {
+         admin-state disable
      }
+     interface ethernet-1/2 {
+         description updated
+     }
```

Example: diff command with single argument

The following shows a diff with a single argument. In this example, the comparison occurs to and from a factory configuration.

```
--{ * candidate shared default }--[  ]--
# diff factory
      acl {
+         acl-filter allow_sip_dip type ipv4 {
+             subinterface-specific output-only
+             entry 10 {
+                 match {
+                     ipv4 {
+                         destination-ip {
+                             prefix 100.1.2.2/32
+                         }
+                         source-ip {
+                             prefix 100.1.1.1/32
+                         }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
+                     }
+                 }
+                 action {
+                     accept {
+                     }
+                 }
+             }
+             entry 100 {
+                 action {
+                     accept {
+                     }
+                 }
+             }
+         }
      }
```

Example: diff between two checkpoints

The following shows an example where the comparison occurs between two checkpoints.

```
--{ candidate shared default }--[  ]--
# save checkpoint name current
/system:
    Generated checkpoint '/etc/opt/srlinux/checkpoint/checkpoint-0.json' with name
 'current' and comment ''

--{ candidate shared default }--[  ]--
# interface ethernet-1/1 description "changed-in-next-checkpoint"
--{ candidate shared default }--[  ]--
# commit stay
All changes have been committed. Starting new transaction.
--{ candidate shared default }--[  ]--
# save checkpoint name newer
/system:
    Generated checkpoint '/etc/opt/srlinux/checkpoint/checkpoint-0.json' with name 'newer'
 and comment ''

--{ candidate shared default }--[  ]--
# diff checkpoint newer checkpoint current
      interface ethernet-1/1 {
-         description changed-in-next-checkpoint
+         description dut1-dut2-1
      }
```

Related topics

NETCONF

3.3.5.1 diff netconf-rpc

Procedure

The **diff** command supports a NETCONF RPC format that, when cut and pasted into a NETCONF session, will result in the same candidate configuration (given the same initial configuration) that is in the current candidate configuration. This format is especially useful when using NETCONF RPCs for automation.

The **diff netconf-rpc** command operates from the present working context that the command runs from. The output takes the form of an <edit-config> RPC.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Note: For the <edit-config> RPC to be valid, it will always output the data back to the root; as a result, the **with-context** option can be provided but may produce an identical output to when it is not provided.

The **diff netconf-rpc** command can be run on either the SR Linux YANG modelled configuration changes or the OpenConfig modelled configuration changes, but the command cannot run on both configuration models at the same time.

Each execution of the **diff netconf-rpc** command must insert a `message-id` into the RPC. In order to avoid potential issues, the `message-id` generated for each command run is different.

The following examples show sample **diff** and **diff netconf-rpc** outputs.

Example: Standard diff output from root

```
--{ * candidate shared default }--[  ]--
A:srl2# diff
      acl {
          system-filter {
              ipv4-filter {
+                 entry 1 {
+                     description example
+                 }
              }
          }
      }
      system {
          name {
+             host-name example
          }
      }
```

Example: Using the diff netconf-rpc command from root

```
--{ * candidate shared default }--[  ]--
A:srl2# diff netconf-rpc
<rpc message-id="0" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
    <target><candidate/></target>
    <config>
        <acl xmlns="urn:nokia.com:srlinux:acl:acl">
            <system-filter>
                <ipv4-filter>
                    <entry>
                        <sequence-id>1</sequence-id>
                        <description>example</description>
                    </entry>
                </ipv4-filter>
            </system-filter>
        </acl>
        <system xmlns="urn:nokia.com:srlinux:general:system">
            <name xmlns="urn:nokia.com:srlinux:chassis:system-name">
                <host-name>example</host-name>
            </name>
        </system>
    </config>
</rpc>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: Using the diff netconf-rpc command from the `/system/name` path

```
--{ * candidate shared default }--[ system name ]--
A:srl2# diff netconf-rpc
<rpc message-id="0" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
    <target><candidate/></target>
    <config>
        <system xmlns="urn:nokia.com:srlinux:general:system">
            <name xmlns="urn:nokia.com:srlinux:chassis:system-name">
                <host-name>example</host-name>
            </name>
        </system>
    </config>
</rpc>
```

3.3.6 Using the copy command

Procedure

Use the **copy** command to copy a specific context to another context. For example, you can use this command to copy a container or any configuration hierarchy and give it another name (such as an interface).

The **copy** command merges the existing context instead of replacing it. If a replace is required, you can delete the target node first. When using the **copy** command, matching fields populate to the destination node; fields that do not match are ignored. The use of ranges for both source and destination is permitted.

Use this command in candidate mode. Nokia also recommends that you use the **diff** command to verify changes when using complex copy operations such as multiple ranges in both the source and destination.

Example: Copy context of one interface to another

The following shows the context of two existing interfaces (ethernet-1/1 and ethernet-2/1):

```
--{ candidate shared default }--[  ]--
# info with-context interface ethernet-1/1
    interface ethernet-1/1 {
        description dut1-dut2-1
        ethernet {
            aggregate-id lag1
        }
    }
--{ candidate shared default }--[  ]--
# info with-context interface ethernet-2/1
    interface ethernet-2/1 {
        description dut2-dut4-1
        subinterface 1 {
            ipv4 {
                admin-state enable
                address 1.2.4.2/24 {
                }
            }
            ipv6 {
                admin-state enable
                address 3ffe:0:2:4::2/64 {
                }
            }
        }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
    }
```

To copy the context of ethernet-1/1 to ethernet-2/1, enter the following:

```
--{ candidate shared default }--[  ]--
# copy interface ethernet-1/1 to interface ethernet-2/1
--{ * candidate shared default }--[  ]--
# diff
      interface ethernet-2/1 {
-         description dut2-dut4-1
+         description dut1-dut2-1
          ethernet {
+             aggregate-id lag1
          }
      }
```

Example: Copy matching fields only

The following shows an example where only some fields match. Fields that do not match are ignored:

```
--{ +* candidate shared default }--[  ]--
# info with-context acl acl-filter cpm type ipv4 entry 10
    acl {
        acl-filter cpm type ipv4 {
            entry 10 {
                description "cpm-filter description"
                match {
                    ipv4 {
                        protocol icmp
                        icmp {
                            type dest-unreachable
                            code [
                                0
                                1
                                2
                                3
                                4
                                13
                            ]
                        }
                    }
                }
                action {
                    accept {
                    }
                }
            }
        }
    }

--{ * candidate shared default }--[  ]--
# copy acl acl-filter cpm type ipv4 entry 10 to interface ethernet-2/1
--{ +* candidate shared default }--[  ]--
# diff
+     interface ethernet-2/1 {
+         description "cpm-filter description"
+     }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: Copy with ranges

The following shows an example that uses ranges. The system expands both the source and destination to determine which set has the largest number of keys, and copies one-to-one until the larger of the two ranges finishes.

```
--{ * candidate shared default }--[  ]--
# copy interface ethernet-1/{1,2} to interface ethernet-{2,3}/{1..10}
--{ * candidate shared default }--[  ]--
# diff
      interface ethernet-2/1 {
-         description dut2-dut4-1
+         description dut1-dut2-1
+         ethernet {
+             aggregate-id lag1
+         }
      }
      interface ethernet-2/2 {
-         description dut2-dut3-1
+         description "second description"
+         mtu 9000
+     }
+     interface ethernet-2/3 {
+         description dut1-dut2-1
+         ethernet {
+             aggregate-id lag1
+         }
      }

+     interface ethernet-3/1 {
+         description dut1-dut2-1
+         ethernet {
+             aggregate-id lag1
+         }
+     }
+     interface ethernet-3/2 {
+         description "second description"
+         mtu 9000
+     }
+     interface ethernet-3/3 {
+         description dut1-dut2-1
+         ethernet {
+             aggregate-id lag1
+         }
+     }
+     interface ethernet-3/4 {
+         description "second description"
+         mtu 9000
+     }

+     interface ethernet-3/9 {
+         description dut1-dut2-1
+         ethernet {
+             aggregate-id lag1
+         }
+     }
+     interface ethernet-3/10 {
+         description "second description"
+         mtu 9000
+     }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.3.7 Using the replace command

Procedure

Use the **replace** command to replace a source context with a destination context (including all of its children). Both the source and destination can be text. This command can also be used to create a destination that does not currently exist by creating a new key that uses the source configuration.

The **replace** command has the following usage: **replace** *<original text>* **with** *<replacement text>*

Use this command in candidate mode.

Example: replace network-instance

The following example replaces all instances of the "mgmt" network-instance with a "test" network-instance:

```
--{ candidate shared default }--[  ]--
# replace "network-instance mgmt" with "network-instance test"
--{ * candidate shared default }--[  ]--
# diff      system {
          ssh-server mgmt {
-             network-instance mgmt
+             network-instance test
          }
          grpc-server mgmt {
-             network-instance mgmt
+             network-instance test
          }
          json-rpc-server {
-             network-instance mgmt {
+             network-instance test {
              }
          }
      }
```

Example: replace interface

The following example shows a command line that can be used to replaces all instances of "interface lag3" or "aggregate-id lag3" with "interface lag2" or "aggregate-id lag2":

```
--{ candidate shared default }--[  ]--
# replace "(interface |aggregate-id )lag3" with \1lag2
```

Example: replace with environment alias

The **replace** command can also be used with the **environment alias** command so that multiple arguments can be used. In the following, an alias named ‘ifrename’ is created to replace the name of an interface:

```
--{ candidate shared default }--[  ]--
# environment alias ifrename "replace \"(interface |aggregate-id ){}\b\" with \1{} /"
```

When the alias is created, the alias with the arguments is entered.

```
--{ candidate shared default }--[  ]--
# ifrename lag2 lag9
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Related topics

Configuring command aliases

3.3.8 Displaying configuration details

Procedure

Use the **info** command to display the configuration. From the root context, enter the **info** command to display the entire configuration. If you include a context with the **info** command, or enter the **info** command within a specified context, the output is limited to the configuration for that context. Alternatively, you can use the **with-context** option to display the full configuration from the root, regardless of the context.

Example: info from root context

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

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

Example: info for specific context

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

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

--{ candidate }--[  ]--
```

Example: info within specific context

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

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


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
--{ candidate }--[ system lldp ]--
```

Example: info within specific context with JSON-formatted output

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

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

Example: info with full context

Use the **with-context** option to display the full context of the configuration. The full context is displayed whether the command is entered from the root context or from within a specified context.

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

Example: info detail

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

```
--{ candidate shared default}--[ system lldp ]--
# info detail
    admin-state enable
    hello-timer 600
    hold-multiplier 4
    management-address mgmt0.0 {
        type [
            IPv4
        ]
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
    }
    interface mgmt0 {
        admin-state disable
    }
```

Example: info flat

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

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

Example: info depth

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

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

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

3.3.9 Displaying the command tree hierarchy

Procedure

Use the **tree** command to display the tree hierarchy for all available nodes you can enter. Entering the **tree** command from the root context displays the entire tree hierarchy. Entering the command from a context limits the display to the nodes under that context. Use this command in candidate or running mode.

Example: tree command

```
--{ candidate shared default}--[ ]--
# tree
<root>
acl
+-- egress-mac-filtering
+-- acl-filter
|  +-- description
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
|  +-- subinterface-specific
|  +-- statistics-per-entry
|  +-- entry
|     +-- description
|     +-- match
|     |  +-- l2
|     |  |  +-- ethertype
|     |  |  +-- destination-mac
|     |  |  |  +-- address
|     |  |  |  +-- mask
|     |  |  +-- source-mac
|     |  |  |  +-- address
|     |  |  |  +-- mask
.
.
.
.
```

3.3.10 Displaying the state of the configuration

Procedure

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

Example: info from state command

To display state information for a specified context from modes that are not state mode:

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

Example: info command from state mode

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

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


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
    }
--{ state }--[  ]--
```

3.3.11 Executing configuration statements from a file

Procedure

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

Example: Execute configuration from a file

The following example executes a configuration from a specified file:

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

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

Lines starting with a number sign (#) are ignored; if the lines contain an opening apostrophe (') or quotation mark (''), the next lines until the corresponding closing character are ignored.

3.3.12 Collecting technical support data

Procedure

To collect technical support data, use the **tech-support** command in any configuration mode.

If you need more control over the information included in technical support packages, use the **tech-support exclude-binaries** command to generate a technical support package that excludes binary and proprietary content. The system creates a technical support package with no encryptions that functions identically to a standard package but with the following exclusions:

- IDB database content
- all application data (app_data)
- other proprietary data not intended for customer exposure

You can access the generated file from the bash shell as shown in Accessing technical support file from bash shell.

Example: Collecting technical support data

The following example generates technical support files including collection of logs, core files, and any application-specific debugs that developers deem necessary to debug issues, and a dump of IDB content.

```
--{ candidate shared default}--[  ]--
# tech-support
Waiting 5 seconds for apps to dump the reports in /tmp/admintech-report-2019_06_19_20_26_
05.zip
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
Finished collecting in 5s
Admin tech report has been generated at /tmp/admintech-report-2019_06_19_20_26_05.zip
--{ candidate }--[  ]--
```

Example: Excluding binaries

The following example generates a technical support package that excludes binaries.

```
--{ candidate shared default }--[  ]--
# tech-support exclude-binaries
Waiting for apps to dump reports for /tmp/tech-support-20250916_185439-srl1.zip
Finished collecting state information in 10 sec

sr_sys_snapshot.sh: Local snapshot ....
sr_sys_snapshot.sh: Local snapshot done in 2 sec
sr_sys_snapshot.sh: Finished: snapshot captured successfully
```

Example: Accessing technical support file from bash shell

The following example show how to access technical support file from bash shell

```
--{ running }--[  ]--
# bash
[root@3-node_srlinux-A /]# cd /tmp
[root@3-node_srlinux-A tmp]# ls -l
total 6012
-rw-r--r-- 1 root root 6110160 Jun 19 20:26 admintech-report-2019_06_19_20_26_05.zip
[root@3-node_srlinux-A tmp]# exit
logout
--{ running }--[  ]--
```

3.4 CLI output formatting and filtering

There are multiple ways to format and filter CLI output, including the following options:

- Specifying the display output format as plain text, a plain text table, JSON, XML, or YAML
- Matching the patterns in the output using Linux tools such as **grep**
- Filtering the output using SR Linux tools such as **eql** or **filter**
- Directing output to a specified file in a specified format

3.4.1 Using Linux command output modifiers

Procedure

You can pipe output from SR Linux CLI commands to standard Linux tools such as **grep**, **more**, **head**, and **tail**.

Example: show interface | grep

The following example pipes the output of the **show interface** command to **grep** so that only lines with `mgmt0` appear in the output:

```
--{ running }--[  ]--
# show interface | grep mgmt0
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
mgmt0 is up, speed None, type None
  mgmt0.0 is up
```

Example: show interface | more

The following example pipes the output of the **show interface** command to **more** to display one page of output at a time:

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

Example: show interface | head

The following example pipes the output of the **show interface** command to **head** to display only the first 8 lines:

```
--{ running }--[  ]--
# show interface | head --lines 8
=============================================================================================
ethernet-1/10 is up, speed 100G, type 100GBASE-CR4 CA-L
  ethernet-1/10.1 is up
    Encapsulation: null
    IPv4 addr    : 192.35.1.0/31 (static)
    IPv6 addr    : 2001:192:35:1::/127 (static, preferred)
    IPv6 addr    : fe80::22e0:9cff:fe78:e2ea/64 (link-layer, preferred)
---------------------------------------------------------------------------------------------
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.4.2 Using the filter output modifier

Procedure

You can use the **filter** output modifier to limit the output of **info**, **list**, **show**, and **watch** commands. Filter options include:

- **node** - Defines a specific node to filter on
- **depth** - Filters out sub-nodes that are deeper than the specified depth
- **fields** - Specifies a specific field or fields to filter on
- **keys-only** - Hides all fields
- **non-zero** - Filters integer fields set to 0 and empty strings

Command output is piped to the **filter** command with the following usage: **filter** [<*node*>] [**depth** <*value*>] [**fields** <*value*>] **[keys-only] [non-zero]**. All options are evaluated together regardless of the order that they are entered with other output modifiers.

Example: Filter info from state command

The following example directs the output of the **info from state** command to filter on the Ethernet node.

```
--{ running }--[  ]--
# info with-context from state interface ethernet-1/1 | filter ethernet
    interface ethernet-1/1 {
        description dut1-dut2-1
        admin-state enable
        mtu 9232
        loopback-mode false
        ifindex 54
        oper-state up
        last-change "an hour ago"
        linecard 1
        forwarding-complex 0
        vlan-tagging false
        ethernet {
            port-speed 10G
            hw-mac-address 00:01:02:FF:00:00
            flow-control {
                receive false
            }
            statistics {
                in-mac-pause-frames 0
                in-oversize-frames 0
                in-jabber-frames 0
                in-fragment-frames 0
                in-crc-error-frames 0
                out-mac-pause-frames 0
                in-64b-frames 0
                in-65b-to-127b-frames 0
                in-128b-to-255b-frames 0
                in-256b-to-511b-frames 0
                in-512b-to-1023b-frames 0
                in-1024b-to-1518b-frames 0
                in-1519b-or-longer-frames 0
                out-64b-frames 0
                out-65b-to-127b-frames 0
                out-128b-to-255b-frames 0
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
                out-256b-to-511b-frames 0
                out-512b-to-1023b-frames 0
                out-1024b-to-1518b-frames 0
                out-1519b-or-longer-frames 0
            }
        }
    }
```

Example: Apply filter to a specific field

Specify the field option to further filter the output to a specific field. For example:

```
--{ running }--[  ]--
# info with-context from state interface ethernet-1/1 | filter fields ethernet/port-speed
    interface ethernet-1/1 {
        ethernet {
            port-speed 10G
        }
    }
```

Example: Apply non-zero filter

The following example shows how you can use the **non-zero** option to display only non-zero values to eliminate non-useful information. The use of the non-zero option only filters integer fields that are currently set to 0 and empty strings.

```
--{ running }--[  ]--
# info with-context from state interface ethernet-1/* statistics | filter non-zero
    interface ethernet-1/1 {
        statistics {
            in-octets 1106761
            in-unicast-packets 1592
            in-broadcast-packets 32
            in-multicast-packets 10736
            in-error-packets 5
            out-octets 2859625
            out-unicast-packets 31576
            out-broadcast-packets 18
            out-multicast-packets 243
            carrier-transitions 3
        }
    }
    interface ethernet-1/2 {
        statistics {
            in-octets 1399961
            in-unicast-packets 129
            in-broadcast-packets 11
            in-multicast-packets 14772
            in-error-packets 1
            out-octets 2474023
            out-unicast-packets 26126
            out-broadcast-packets 19
            out-multicast-packets 173
            carrier-transitions 3
        }
    }
    interface ethernet-1/20 {
        statistics {
            in-octets 1443012
            in-unicast-packets 350
            in-broadcast-packets 12
            in-multicast-packets 15639
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
            in-error-packets 3
            out-octets 2319629
            out-unicast-packets 25558
            out-broadcast-packets 17
            out-multicast-packets 199
            carrier-transitions 3
        }
    }
```

3.4.3 Using the eql output modifier

About this task

The **eql** output modifier uses the EDA Query Language (EQL, see "EDA query language" in the *EDA User Guide*) to filter SR Linux **info** or **monitor once** command outputs with EQL queries.

Procedure

Use the following guidelines and commands to construct EDA queries:

- The query string must contain an ordered set of fields, operators, values, and keywords enclosed by double quotations (").
- You may have a condition without a query string on a field, for example, `info from state interface * | eql "mtu"`. The condition is only `True` if the value is non-zero for integers or non-zero length for strings.
- The EQL statement follows the rules of the `where` clause in EQL with the assumption that an initial parenthesis set is already provided. After this instance, parentheses may be used normally to group statements together.
- Field names are unquoted.
- Values are quoted with single quotations when they are strings and unquoted when they are integers. For example, `oper-state = 'up'` and `ifindex = 49150`. Integers can also be quoted in single quotations to be processed as a string, but strings must always be quoted in single quotations.
- Use `()` and keywords `AND` and `OR` to group multiple criteria. For example, `(oper-state = 'down' and mtu = 1500) or oper-state = 'up'`. Keywords may be capitalized. For example, both `and` and `AND` are valid.
- You can query any child field in a container, but you cannot navigate into lists unless it is within the same application. The syntax is `<parent>.<child>`. For example, to query the field `error-count` in the `statistics` container, the query should could read `statistics.error > 0`.
- Evaluations against a field yield `True` for equal, `False` for not equal, and `Null` if the field is unset with the following logical conditions:
  - `not Null` is `Null`
  - `X and Null` is `Null`
  - `X or Null` is `X`

  For example, assuming `oper-state` has the value `up`:
  - A query would return `True` and return the interface: `oper-state = 'up'`.
  - A query would return `False` and not return the interface: `oper-state != 'up'`.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Assuming `oper-state` has the value `down`:

- A query would return `False` and not return the interface: `oper-state = 'up'`.
- A query would return `True` and return the interface: `oper-state != 'up'`.

Assuming the `oper-state` value is not set:

- A query would return `Null` and not return the interface: `oper-state = 'up'`.
- A query would return `Null` and not return the interface: `oper-state != 'up'`.

- The following operators are valid:
  - `=`
  - `!=`
  - `<=`
  - `>=`
  - `>`
  - `<`
  - `~`
  - `is not set`
  - `is set`
  - `in`
  - `not in`

  The `is set` operator is used to check if a field is set to a specific value and the `is not set` operator is used to check if a field is not set or has no default value.
  The `in` operator takes an array of values. For example, `oper-state in ['up', 'down']`. The `in` operator also takes the values of another field if that field is a leaf list.

The SR Linux EQL implementation has the following differences compared to the full implementation available in EDA:

- Only conditions from the `where` clause are supported.
- A condition may not cross an application boundary.
- **show** commands do not support EQL filtering.
- Queries are run once when the command is executed, and do not support streaming subscriptions like `SAMPLE` or `ON_CHANGE`.

Example: Modify info from state output

The following example modifies the output of the **info from state** command to display interfaces that are operationally down and are administratively enabled.

```
info from state interface * | eql "oper-state = 'down' AND admin-state = 'enable'"
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.4.4 Using the jq output modifier

About this task

The **jq** output modifier (see https://jqlang.github.io/jq/) is a JSON processor that can manipulate SR Linux **date**, **diff**, **info**, **list**, **references**, **show**, and **watch** command output, provided the output is displayed in JSON format (using the **| as json** output modifier). You can use **jq** to filter, extract, combine, and modify the JSON output. You can also use piping to combine **jq** arguments in various ways, such as feeding the results of one filter into another, or collecting the output into an array. The **jq** processor is supported as a CLI plug-in that is enabled by default.

Procedure

To modify the **show** or **info** output with the **jq** output modifier, include the **| as json** output modifier in the command, followed by **| jq** and one or more **jq** arguments.



Note: SR Linux uses double quotes (**"**) rather than single quotes (**'**) to define **jq** filter expressions in the jq output modifier. Therefore, if a **jq** expression contains double quotes, they must be escaped using a backslash (**\"**).

Example: Modify info from state output

The following example modifies the output of the **info from state** command to list applications that have a status of **waiting-for-config**.

```
--{ running }--[  ]--
# info with-context from state system app-management application * state | as json | jq
 ".system.\"app-management\".application[] | select(.state == \"waiting-for-config\")
 | .name"
"dhcp_relay_mgr"
"dhcp_server_mgr"
"ethcfm_mgr"
"event_mgr"
"fhs_mgr"
"gribi_server"
"igmp_mgr"
"isis_mgr"
"l2_proxy_arp_nd_mgr"
"l2_static_mac_mgr"
"label_mgr"
"ldp_mgr"
"macsec_mgr"
"mirror_mgr"
"mpls_mgr"
"oam_mgr"
"oc_mgmt_server"
"ospf_mgr"
"p4rt_server"
"pim_mgr"
"segrt_mgr"
"static_route_mgr"
"te_mgr"
"tepolicy_mgr"
"twamp_mgr"
"vrrp_mgr"
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.4.5 Using the yq output modifier

About this task

The **yq** output modifier (see https://mikefarah.gitbook.io/yq/) is a YAML processing tool inspired by **jq** that can manipulate SR Linux **date**, **diff**, **info**, **list**, **references**, **show**, and **watch** command output, provided the output is displayed in YAML format (using the **| as yaml** output modifier). Similar to **jq**, you can use **yq** to filter, extract, combine, and modify the YAML output. You can also use piping to combine **yq** arguments in various ways, such as feeding the results of one filter into another or collecting the output into an array. The **yq** processor is supported as a CLI plug-in that is enabled by default. You can also invoke **yq** directly using the **bash cat** *<filename>.yml* **| yq** *<arguments>* command.

Procedure

To modify the **show** or **info** output with the **yq** output modifier, include the **| as yaml** output modifier in the command, followed by **| yq** and one or more **yq** arguments.

Note: SR Linux uses double quotes (**"**) rather than single quotes (**'**) to define **yq** filter expressions in the **yq** output modifier. Therefore, if a **yq** expression contains double quotes, they must be escaped using a backslash (**\"**).

Example: Modify info output

The following example modifies the output of the **info** command to display interfaces that have an IP MTU greater than 1500:

```
info / interface * | as yaml | yq -C e ".interface[] | select(.subinterface[]?.["ip-mtu"]
 > 1500)"
```

Figure 1: Modify info output

```
--{ + running }--[  ]--
A:ixr6e# info / interface * | as yaml | yq -C e ".interface[] | select(.subinterface[]?.[\"ip-mtu\"] > 1500)"
name: ethernet-1/1
admin-state: enable
subinterface:
  - index: 0
    admin-state: enable
    ip-mtu: 9000
    ipv4:
      admin-state: enable
      address:
        - ip-prefix: 192.168.1.1/24
name: ethernet-1/2
admin-state: enable
subinterface:
  - index: 0
    admin-state: enable
    ip-mtu: 9000
    ipv4:
      admin-state: enable
      address:
        - ip-prefix: 192.168.1.2/24
--{ + running }--[  ]--
```

sc0169

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.4.6 Using the as output modifier

About this task

The **as** output modifier displays SR Linux **baseline diff**, **date**, **diff**, **info**, **list**, **references**, **show**, and **watch** command outputs in plain text (default), a plain text table, JSON, XML, or YAML format. You can set the default output format with the **environment output-format** command. For more information, see Configuring the output format.

Example: Plain text output (default)

```
--{ candidate shared default }--[ interface mgmt0 ]--
# info
    description management
    admin-state enable
    subinterface 0 {
        admin-state enable
        ip-mtu 1500
        ipv4 {
            admin-state enable
            dhcp-client {
            }
        }
        ipv6 {
            admin-state enable
            dhcp-client {
            }
        }
    }
```

Note: The **| as text** output modifier can also be used to display plain text output.

Example: Plain text table output

```
--{ state }--[ interface mgmt0 ]--
# info depth 0 | as table
+--------------------+-------------+------------+-----------------------------------+
|        Name        | Admin-state | Oper-state |          Oper-down-reason         |
+====================+=============+============+===================================+
| mgmt0              | enable      | up         |                                   |
+--------------------+-------------+------------+-----------------------------------+
```

Example: JSON output

```
--{ candidate shared default }--[ interface mgmt0 ]--
# info | as json
{
    "name": "mgmt0",
    "description": "management",
    "admin-state": "enable",
    "subinterface": [
        {
            "index": 0,
            "admin-state": "enable",
            "ip-mtu": 1500,
            "ipv4": {
                "admin-state": "enable",
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
                "dhcp-client": {
                }
                },
                "ipv6": {
                    "admin-state": "enable",
                    "dhcp-client": {
                    }
                }
            }
        ]
    }
```

Example: XML output

```
--{ candidate shared default }--[ interface mgmt0 ]--
# info | as xml
<description xmlns="urn:nokia.com:srlinux:chassis:interfaces">management</description>
<admin-state xmlns="urn:nokia.com:srlinux:chassis:interfaces">enable</admin-state>
<subinterface xmlns="urn:nokia.com:srlinux:chassis:interfaces">
    <index>0</index>
    <admin-state>enable</admin-state>
    <ip-mtu>1500</ip-mtu>
    <ipv4>
        <admin-state>enable</admin-state>
        <dhcp-client xmlns="urn:nokia.com:srlinux:chassis:interfaces-ip-dhcp"/>
    </ipv4>
    <ipv6>
        <admin-state>enable</admin-state>
        <dhcp-client xmlns="urn:nokia.com:srlinux:chassis:interfaces-ip-dhcp"/>
    </ipv6>
</subinterface>
```

Note: The XML output displays a complete and correct XML document that includes all namespaces, namespace definitions, annotations, metadata, operations, and identityref encoding.

Example: YAML output

```
--{ candidate shared default }--[ interface mgmt0 ]--
# info | as yaml
---
name: mgmt0
description: management
admin-state: enable
subinterface:
    - index: 0
    admin-state: enable
    ip-mtu: 1500
    ipv4:
        admin-state: enable
        dhcp-client:
    ipv6:
        admin-state: enable
        dhcp-client:
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.4.7 Directing output to a file

Procedure

You can direct the output of SR Linux CLI commands to a specified file. The output can be saved as text, in table format, or in JSON format.

Example: Direct show interface output to file

The following example directs the output of the **show interface** command to a file. The output is saved in JSON format.

```
--{ running }--[  ]--
# show interface | as json > show_interface.json
```

Use **>** to create a new file with the specified filename; if the file already exists, it is replaced. Use **>>** to append the output to the specified file if it exists.

Example: Access output file in Bash

You can access the file using Bash. For example:

```
--{ * running }--[  ]--
# bash
[bob@3-node-srlinux-A /]# more show_interface.json
{
  "interfaces": [
    {
      "Interface": "ethernet-1/1",
      "subinterfaces": [
        {
          "Subinterface": "ethernet-1/1.1",
          "Oper": "up",
          "IPv4 Addresses": "192.168.11.1/30",
          "IPv6 Addresses": "2001:1::192:168:11:1/126, fe80::1012:5ff:feff:0/64"
        }
      ]
    },
    {
--More--(42%)
```

3.5 CLI environment customization

You can optionally configure the SR Linux CLI environment to change settings such as command aliases, the command prompt, the contents of the bottom toolbar, and the default format for displayed output.

The CLI has two environment configurations:

- **global environment configuration**

  The global environment configuration is set with **system cli environment** commands, which are persistent in the startup configuration file and are loaded from `/etc/opt/srlinux/srlinux.rc` when a user logs in. When the environment settings are committed, they are applied to all active sessions, except for the CLI engine type. To avoid overwriting values in a session, only values that differ from default values are loaded. Logging out and back in ensures that all values from the global environment configuration are applied to the new session.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- **per-user environment**

  The per-user environment configuration is set with **environment** commands, which are not persistent unless saved with the **environment save** command. When the per-user environment is saved, it is automatically loaded each time you log in after the global CLI environment settings are loaded.

The environment commands described in the examples in the following sections use the per-user **environment** command format, but can also be set with the global **system cli environment** commands which have a similar syntax.

Note: The `/etc/opt/srlinux/srlinux.rc` file must not be edited, because it is generated when the global environment is committed.

3.5.1 Displaying the environment configuration

Example: Display the global CLI environment settings

You can display the global CLI environment settings with the **info detail** command.

```
--{ running }--[ ]--
# info detail with-context system cli environment
    system {
        cli {
            environment {
                basic-prompt "\\n--{{ {banner}{startup_config_state}{modified_flags}
{short_yang_models}{mode_and_session} }}--[ {pwc} ]--\\n{short_redundancy}{hw_slot}:{aaa_
user}@{host}# "
                bottom-toolbar "{banner}{startup_config_state}{commit_confirmed_with_
remaining_time}|{time_24}"
                complete-on-enter false
                complete-on-space false
                complete-on-tab true
                key-completer-limit 100
                output-format text
                prompt "\\n--{{ {modified_flags}{short_yang_models}{mode_and_session} }}--
[ {pwc} ]--\\n{short_redundancy}{hw_slot}:{aaa_user}@{host}# "
                yang-models srl
                cli-engine {
                    completion-ignore-case false
                    completion-display popup
                    completion-type smart
                    history-filename ~/.srlinux_history
                    max-history-items 1000
                    refresh-interval 5
                    type advanced
                    vi-editing-mode false
                }
            }
        }
    }
```

Example: Display the per-use CLI environment settings

You can display the per-user CLI environment settings with the **environment show** command.

```
--{ running }--[  ]--
# environment show
[alias]
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
[basic-prompt]
value = "\n--{{ {banner}{startup_config_state}{modified_flags}{short_yang_models}{mode_
and_session} }}--[ {pwc} ]--\n{short_redundancy}{hw_slot}:{aaa_user}@{host}# "

[bottom-toolbar]
value = "{banner}{startup_config_state}{commit_confirmed_with_remaining_time}|{time_24}"

[cli-engine]
type = "advanced"
max-history-items = 1000
history-filename = "~/.srlinux_history"
completion-type = "smart"
completion-ignore-case = false
completion-display = "popup"
vi-editing-mode = false
refresh-interval = 5.0

[enter-completion]
enabled = false

[key-completer-limit]
limit = 100

[network-instance]
value = ""

[output-display-format]
value = "text"

[output-modifier-alias]

[pagination]
value = ""

[prompt]
value = "\n--{{ {modified_flags}{short_yang_models}{mode_and_session} }}--[ {pwc} ]--\
n{short_redundancy}{hw_slot}:{aaa_user}@{host}# "

[session-idle-timeout]
value = ""

[space-completion]
enabled = false

[tab-completion]
enabled = true

[yang-models]
value = "srl"
```

3.5.2 Managing environment settings

Procedure

You can save your current per-user CLI environment settings to `~/.srlinuxrc` using the **environment save home** command, and you can load them using the **environment load home** command. These settings are automatically loaded each time you log in after the global CLI environment settings are loaded.

Alternatively, you can save the CLI environment settings to a different file using the **environment save file** command and you can manually load them from the file using the **environment load file** command.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

You can revert your CLI environment settings for the current CLI session to the system default settings using the **environment delete** command. You can also revert your CLI environment settings to a specific setting by specifying it after the **environment delete** command. This command does not delete your saved environment settings or the global environment settings.

The username `srlinux` is used in the following examples.

Example: Save your CLI environment settings to their default location

The following example saves your current CLI environment settings:

```
--{ running }--[  ]--
# environment save home
Saved configuration to /home/srlinux/.srlinuxrc
```

Example: Load your default CLI environment settings

The following example loads your default CLI environment settings:

```
--{ running }--[  ]--
# environment load home
Loaded configuration from /home/srlinux/.srlinuxrc
```

Example: Save your CLI environment settings to a file

The following example saves your current CLI environment settings to a file:

```
--{ running }--[  ]--
# environment save file env.cfg
Saved configuration to env.cfg
```

Example: Load your CLI environment settings from a file

The following example loads your CLI environment settings from a file:

```
--{ running}--[  ]--
# environment load file env.cfg
Loaded configuration from env.cfg
```

3.5.3 Configuring command aliases

Procedure

As a shortcut for entering commands in the CLI, you can configure command aliases using the **environment alias** command. The alias can include one or more CLI command keywords and arguments.

Command aliases can also be configured to output modifiers using the **environment output-modifier-alias** command.

Command alias names may contain any printable UTF-8 characters, but the following characters are not supported:

- angle brackets (<, >)
- double quotation marks (")
- newline characters

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- single quotation marks (')
- space — in output modifier aliases
- square brackets ([,])
- tab
- vertical bar (|)

The following characters in command alias values must be escaped with a backslash (\):

- backslash (\) — in the global CLI environment configuration
- double quotation marks (")

Example: Set alias for info interface command

The following example configures the alias **display-interface** for the SR Linux command **info interface** <*name*> **subinterface** <*index*> **| as table**:

```
--{ running }--[  ]--
# environment alias "display-interface" "info / interface {} subinterface {subinterface} |
 as table"
```

In the example, the **display-interface** alias consists of the keywords **info interface**, arguments to specify an interface name and subinterface number, and keywords to display the output as a table. The alias name and aliased command are each enclosed in quotes. The arguments are enclosed in braces (**{ }**). The argument **{}** creates an optional unnamed variable for the interface name, and the argument **{subinterface}** creates an optional parameter named `subinterface`.

When you enter the alias at the CLI prompt, the output of the aliased command is displayed. For example:

```
--{ running }--[  ]--
# display-interface ethernet-1/1 subinterface 1
+---------------------+-------+-------------+
|      Interface      | Index | Admin-state |
+=====================+=======+=============+
| ethernet-1/1        |     1 | enable      |
+---------------------+-------+-------------+
```

If you omit the optional parameters for the interface and subinterface names, they are treated as wildcards. For example:

```
--{ running }--[  ]--
# display-interface
+---------------------+-------+-------------+
|      Interface      | Index | Admin-state |
+=====================+=======+=============+
| ethernet-1/1        |     1 | enable      |
| ethernet-1/2        |     1 | enable      |
| lo0                 |     1 | enable      |
| mgmt0               |     0 | enable      |
+---------------------+-------+-------------+
```

Example: Set an output modifier alias

The following example configures the alias **match** to the SR Linux **grep** command.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

You can enter output modifier aliases entered after **|** in a command. Output modifier aliases can represent a complete command or part of a target command. To display output modifier aliases use the **info | ??** command.

```
--{ running }--[  ]--
# environment output-modifier-alias match grep

--{ running }--[  ]--
# environment show
[output-modifier-alias]
match = "grep"

--{ running }--[  ]--
# info | match host-name
host-name node-2
```

3.5.4 Configuring command completion

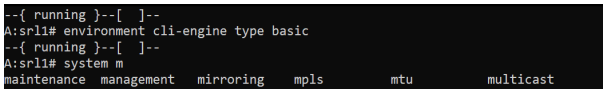
About this task

When you enter a partial command at the CLI prompt and press **Tab** or **Spacebar**, SR Linux completes the command if no other command at that level starts with those letters.

If multiple commands at that level start with the characters you typed, SR Linux displays a list of options that can complete the command.

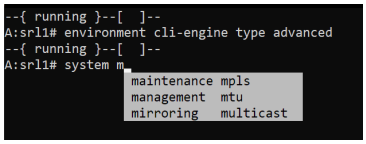
- When the basic CLI engine type is configured, SR Linux displays the command options as a non-selectable list. For example:

  Figure 2: Basic CLI engine command completion



- When the advanced CLI engine type is configured, SR Linux displays a popup box with possible commands that are valid at that level. You can press the up or down arrows to select a command in the list. For example:

  Figure 3: Advanced CLI engine command completion



The command options that appear when you press **Tab** or **Spacebar** depend on the SR Linux command completion type setting, which are outlined in the following table.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Table 7: Command completion type settings

Command completion type	Definition
prefix	The displayed options start with the letters you typed. This is the only valid command completion type for the basic CLI engine.
smart	The displayed options contain the letters you typed. Preference is given to the following: • Options that start with the letters you typed. • Multi-word options where each word in the option starts with a letter you typed; for example, typing **ni** displays `network-instance` as an option. • Options that contain the letters you typed; for example, typing **inst** displays `network-instance` as an option. • Options that exist only at the current CLI level, rather than those that exist at every CLI level. This is the default command completion type for the advanced CLI engine.
substring	The displayed options contain the letters you typed as a string within the option name; for example, typing **stat** displays `modules-state`, `netconf-state`, and `statistics` as options.
fuzzy	The displayed options contain the letters you typed as a string, as well as options that are close; for example, typing **ntw** displays `network-instance` as an option.

By default, when you press **Tab** or **Spacebar** to complete the next command level, a popup box appears that displays the available options for that command. Pressing **Enter** also completes the command without a popup box.

The following usage guidelines apply to command completion:

- **Tab** completion — if there is no matching local command, uses the configured completion type and displays global commands
- **Spacebar** completion — if there is no matching local command, uses the configured completion type and displays global commands
- **Enter** completion — uses the **prefix** completion type and completes matching local and global commands together

You can also complete commands by pressing **Alt+/**, which always uses fuzzy completion.

Procedure

To configure the SR Linux command completion type, use the **environment cli-engine completion-type** command.

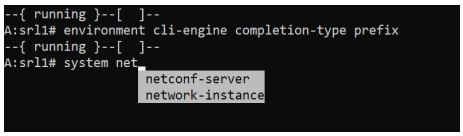
Example: Set the command completion type to prefix

The following example configures SR Linux to use **prefix** as the command completion type.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

In this example, if you type **system net** and press **Tab**, SR Linux displays the command options at the system level that start with `net`.

Figure 4: Prefix command completion type

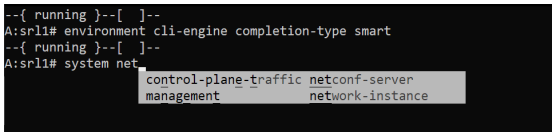


Example: Set the command completion type to smart

The following example configures SR Linux to use **smart** as the command completion type.

In this example, if you type **system net** and press **Tab**, SR Linux displays the command options at the system level that start with `net`, as well as those that include the letters n, e, and t.

Figure 5: Smart command completion type

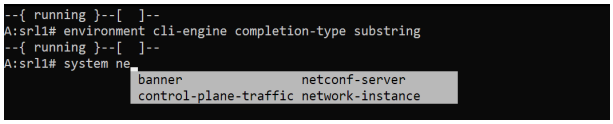


Example: Set the command completion type to substring

The following example configures SR Linux to use **substring** as the command completion type.

In this example, if you type **system ne** and press **Tab**, SR Linux displays the command options at the system level that include the text string `ne`.

Figure 6: Substring command completion type

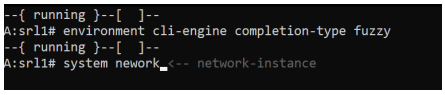


Example: Set the command completion type to fuzzy

The following example configures SR Linux to use **fuzzy** as the command completion type.

In this example, if you type **system nework** and press **Tab**, SR Linux corrects the spelling of the command to **system network**, since **system network-instance** is a valid command at this CLI level.

Figure 7: Fuzzy command completion type



Example: Disable Spacebar command completion

Command completion with **Spacebar** is enabled by default. To disable command completion with **Spacebar**, set the **environment complete-on-space** command to **false**.

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


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
# environment complete-on-space false
```

Example: Disable Enter command completion

Command completion with **Enter** is enabled by default. To disable command completion with **Enter**, set the **environment complete-on-enter** command to **false**.

```
--{ candidate shared default }--[  ]--
# environment complete-on-enter false
```

Example: Disable Tab command completion

Command completion with **Tab** is enabled by default. To disable command completion with **Tab**, set the **environment complete-on-tab** command to **false**.

```
--{ candidate shared default }--[  ]--
# environment complete-on-tab false
```

Example: Ignore the word case

By default, the case of words in completion is considered part of the match criteria. For example, a lower case letter that is entered before **Tab** is pressed will not match a key name starting with an upper case letter. To ignore the word case, set the **environment completion-ignore-case** command to **true**.

```
--{ running }--[  ]--
# environment completion-ignore-case true
```

Example: Limit the number of displayed keys

You can limit the number of keys that are displayed in completion when there are a large number of keys in a list that match, to display a subset of matches. By default, the first 100 keys that match are displayed. To change the limit, set the **environment key-completer-limit** command value to the desired number of keys.

```
--{ running }--[  ]--
# environment key-completer-limit 200
```

Example: Set the command completion display type

You can set the display style of possible commands using the **environment cli-engine completion-display** command.

The default value for **completion-display** is **popup**, where completion displays the available options for that command as a popup box.

When **completion-display** is set to **inline**, the popup box is not displayed. Instead, the command options are displayed as a non-selectable list, like in the basic CLI engine.

```
--{ running }--[  ]--
# environment cli-engine completion-display inline
```

Related topics

Using the CLI command completion

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.5.5 Configuring pagination

About this task

By default, pagination of command output is disabled and the **| more** output modifier must be used to paginate the output. To enable pagination by default, set the **environment pagination** command to **on**.

When pagination is enabled, the **| no-more** output modifier may be used to disable pagination for a single command.

Example: Enable pagination for all commands

```
--{ running }--[  ]--
# environment pagination on
```

Example: Disable pagination for a single command

```
--{ running }--[  ]--
# info | no-more
```

3.5.6 Configuring the bottom toolbar

About this task

By default, the text that appears in the toolbar at the bottom of the terminal window in CLI sessions displays the mgmt_server status banner, startup configuration state, if a confirmed commit is running, and the local time in the following format:

```
environment bottom-toolbar "{banner}{startup_config_state}{commit_confirmed_with_remaining_
time}|{time_24}"
```

For example:

```
[commit confirmed (rollback in 9 minutes)]                              Mon 19:55
```

Procedure

Use the **environment bottom-toolbar ?** command to display the keywords that you can configure in the bottom toolbar.

You can configure the bottom toolbar using the **environment bottom-toolbar** command to include information such as the number of changes made to the configuration and the host name.

Example: Add number of changes and host name to the toolbar

The following example adds the number of changes made to the configuration and the host name to the bottom toolbar.

```
--{ running }--[  ]--
# environment bottom-toolbar "Current mode: {modified_with_change_count}{mode_and_session_
type}| {aaa_user}@{host} ({aaa_session_id})  {time}"
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

In the example, the number of configuration changes is configured with the **{modified_with_change_count}** keyword, and the host name is configured with the **{host}** keyword.

After you enter this command, the bottom toolbar looks like the following:

```
Current mode: *10 candidate shared       admin@3-node-srlinux-A (36)  Wed 10:18PM
```

Example: Set the toolbar refresh interval

By default, the toolbar display refreshes every 5 seconds and when **Enter** is pressed. To configure a refresh interval in seconds, set the value of the **environment cli-engine refresh-interval** command. A value of 0 disables the toolbar refresh.

```
--{ running }--[  ]--
# environment cli-engine refresh-interval 1
```



Note:
A low refresh interval may affect system performance if the bottom toolbar is configured to call compute-intensive Python functions.

3.5.7 Configuring the command history

About this task

The CLI command history records the last 1000 commands executed exactly as they were entered after **Enter** is pressed in `~/.srlinux_history`. You can change the history size with the **environment cli-engine max-history-items** command, and you can change the history filename with the **environment cli-engine history-filename** command.

Example

The following example sets the history size to 50 commands:

```
--{ running }--[ ]--
# environment cli-engine max-history-items 50
```

The following example sets the history filename to `~/.my_history`:

```
--{ running }--[ ]--
# # environment cli-engine history-filename ~/.my_history
```

3.5.8 Configuring vi editing mode

About this task

You can configure the editing mode to use **vi**-style key bindings.

The following actions are supported:

- Enter insert mode, the default mode, with editing commands such as **a**, **A**, **i**, **I**, **Enter** and **Insert**.
- Enter navigation mode, a combination of normal and visual mode, with **Esc**.
- Replace the character at the cursor with **r**.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- Replace multiple character starting at the cursor with **R**.
- Move and edit with standard **vi** commands.
- Edit with **vi** macros and registers.

Example: Enable vi editing mode

The following example enables **vi** editing mode:

```
--{ running }--[ ]--
# environment cli-engine vi-editing-mode true
```

3.5.9 Configuring the engine type

About this task

SR Linux features two versions of the CLI engine: advanced (default) and basic. The advanced CLI engine is enabled by default; it includes the following features:

- Displays a toolbar at the bottom of the terminal window.
- Includes the command auto-complete feature with the **Enter**, **Spacebar**, and **Tab** keys.
- Allows you to select command options by pressing the **Tab** key to display the options in a popup box, then using the arrow keys to select an option.
- Displays descriptions of command options when you press the **?** key.

The basic CLI engine includes the following reduced set of features and is designed to be used primarily by automation workflows that use CLI screen scraping:

- Omits the toolbar at the bottom of the terminal window.
- Includes the command auto-complete feature with the **Enter** and **Tab** keys.
- Displays a plain text list of command options when you press the **Tab** key.
- Displays descriptions of command options when you press the **?** key and press **Enter**, but only for the current context.

Procedure

If necessary, you can use the **environment cli-engine type basic** command to configure SR Linux to use the basic CLI engine instead of the advanced CLI engine for CLI sessions.

Example: Set the CLI engine type to basic

The following example configures the SR Linux to use the basic CLI engine for CLI sessions:

```
--{ running }--[  ]--
# environment cli-engine type basic
```

Related topics

Configuring the bottom toolbar

Using the CLI command completion

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.5.10 Configuring the network instance

About this task

When you enter administrative commands such as **ping** and **traceroute**, you can specify the network instance to be used with the command. If you do not specify a network instance, then the network instance configured in the network instance environment setting is used.

If you do not specify a network instance, or the network instance cannot be inferred from the CLI context, then the *default* network instance is used in a **ping** or **traceroute** command.

Procedure

To configure the network instance environment setting, use the **environment network-instance** command.

Example: ping with no network instance specified

```
--{ running }--[  ]--
# ping 192.168.1.1 -c 3
Using network instance default
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.027 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.030 ms

--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 6165ms
rtt min/avg/max/mdev = 0.027/0.030/0.033/0.005 ms
```

Example: Configure the environment network instance

```
--{ running }--[  ]--
# environment network-instance mgmt
```

Example: Ping using the environment network instance settings

In this example, the network instance environment setting is set to **mgmt**, so network instance *mgmt* is used when the **ping** or **traceroute** command is entered without a network instance name. For example:

```
--{ running }--[  ]--
# ping 192.168.1.1 -c 3
Using network instance mgmt
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.027 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.030 ms

--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 6165ms
rtt min/avg/max/mdev = 0.027/0.030/0.033/0.005 ms
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.5.11 Configuring the output format

Procedure

You can configure the output of CLI commands using the **output-format** command to be displayed as plain text (default), a plain text table, JSON, XML, or YAML format.

Example: Set the output to JSON format

The following example configures CLI command output to be displayed in JSON format.

```
--{ running}--[  ]--
# environment output-format json
```

Subsequent command output is displayed in JSON format by default. For example:

```
--{ running }--[  ]--
# show version
{
  "basic system info": {
    "Hostname": "3-node-srlinux-A",
    "Chassis Type": "7250 IXR-10",
    "Part Number": "Sim Part No.",
    "Serial Number": "Sim Serial No.",
    "System MAC Address": "12:12:02:FF:00:00",
    "Software Version": "v19.11.1",
    "Build Number": "291-g4664705",
    "Architecture": "x86_64",
    "Last Booted": "2019-12-07T00:34:48.942Z",
    "Total Memory": "16396536 kB",
    "Free Memory": "5319448 kB"
  }
}
```

3.5.12 Configuring the prompt

About this task

By default, the SR Linux advanced CLI engine prompt consists of two lines of text and indicates with an asterisk whether the configuration has been modified, displays the current mode and session type, the current CLI context, redundancy information, the user, and the host name of the SR Linux device in the following format:

```
environment prompt "\n--{{ {modified_flags}{short_yang_models}{mode_and_session} }}--[ {pwc} ]-
-\n{short_redundancy}{hw_slot}:{aaa_user}@{host}# "
```

For example:

```
--{ running }--[  ]--
A:admin@srl1#
```

By default, the SR Linux basic CLI engine prompt consists of two lines of text displaying the `mgmt_server` status banner and the startup configuration state. It also indicates with an asterisk whether the

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

configuration has been modified, displays the current mode and session type, the current CLI context, redundancy information, the username, and the host name of the SR Linux device in the following format:

```
environment basic-prompt "\n--{{ {banner}{startup_config_state}{modified_flags}{short_yang_
models}{mode_and_session} }}--[ {pwc} ]--\n{short_redundancy}{hw_slot}:{aaa_user}@{host}# "
```

For example:

```
--{ [SYSTEM REBOOT IN 59 SECONDS (Upgrading software.)] running }--[  ]--
A:admin@srl1#
```

Procedure

Use the **environment prompt ?** command to display the keywords that you can configure in the SR Linux CLI prompt.

You can configure the SR Linux prompt to include information such as session ID of the CLI session, the number of changes made to the configuration, and the current local time.

Example: Add local time and session username to CLI prompt

The following example adds the local time to the SR Linux CLI prompt.

```
--{ running }--[  ]--
# environment prompt "--{{ {modified}{mode_and_session_type} }}--[ {pwc} ]--{time 12}--\
n{user}@{host}# "
```

In the example, the local time is configured with the **{time 12}** keyword. The line break is configured with **\n**.

After you enter this command, the CLI prompt looks like the following:

```
--{ * candidate shared default}--[ acl ]--Wed 03:07PM--
admin@srl1#
```

3.5.13 Configuring the session idle timeout

About this task

The session idle timeout that closes inactive CLI sessions is configured for all users with the **system aaa authentication idle-timeout** command.

To configure a different idle timeout in seconds, set the value of the **environment session-idle-timeout** command. A value of 0 disables the idle timeout so you must log out manually, and an unset value uses the system timeout.

Example: Set the session idle timeout to 600 seconds

The following example sets the session idle timeout to 600 seconds.

```
--{ running }--[ ]--
# environment session-idle-timeout 600
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

3.5.14 Configuring the data model

About this task

By default, the data model used in the CLI is SR Linux (**srl**). To set the default data model to a different one, use the **environment yang-models** command.

Example: Set the default data model to OpenConfig

The following example sets the default data model to OpenConfig.

```
--{ running }--[ ]--
# environment yang-models oc
```

3.5.15 Prompt and bottom toolbar keywords

The information that is displayed in the prompt and in the bottom toolbar can be customized with UTF-8 strings and keywords. Use the following formatting directives to customize the output:

- Enclose keywords with braces (**{}**), for example `{pwc}`.
- Insert newlines with **\n** to create a prompt with multiple lines.
- Display a { with **{{** and a } with **}}**, for example `{{foo}}` displays `{foo}`.
- Right-justify the output after a vertical bar (**|**); the vertical bar is not displayed.

The following table describes keywords that can be used.

Table 8: Prompt and bottom toolbar keywords

Keyword	Definition	Values
aaa_session_id	The incrementing session ID	0 to 4294967295
aaa_user	The username authenticated in the session	A string, for example, `srlinux`
banner	The mgmt_server status banner	• `[ACTIVE CONTROL REBOOT IN` *`remaining time`* `SECONDS (`*`message`*`)]` : the active CPM will reboot • `[OLD RESCUE]` : the rescue configuration file was saved with an older software version • `[OLD STARTUP]` : the startup configuration file was saved with an older software version • `[SYSTEM REBOOT IN` *`remaining time`* `SECONDS`

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Keyword	Definition	Values
		`(`*`message`*`)]` : the system will reboot • `[WARM BOOT]` : a warm boot is in progress • `[YANG LOAD]` : YANG models are loading • `[YANG RELOAD]` : YANG models are reloading
candidate_name	The candidate name	• A string: for example, `default` • No value: when not in a configuration mode
commit_confirmed	If a commit confirmed is running	• `[commit confirmed]` : a commit confirmed is running • No value: a commit confirmed is not running
commit_confirmed_with_ remaining_time	If a commit confirmed is running and the remaining time before an automatic rollback	• `[commit confirmed (rollback in` *`remaining time`*`)]` : a commit confirmed is running, for example 9 `minutes` or `45 seconds` • No value: a commit confirmed is not running
configuration_session_type	The configuration mode type	• `private`: private mode • `private-exclusive`: private exclusive mode • `shared`: shared mode • `shared-exclusive`: shared exclusive mode • No value: when not in a configuration mode
domain	The domain name	• A string: for example, `example.com` • No value: if the domain name is not configured
fqdn	The fully qualified domain name (FQDN)	• A string: for example, `srlinux.example.com`

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Keyword	Definition	Values
		• A string: the host name if the domain name is not configured
host	The host name	A string: for example, `srlinux`
hw_slot	The active CPM slot	• `A`: slot A • `B`: slot B
jspath_pwc	The present working context in JSPath format	A string: for example, `.network-instance{.name== "default"}.protocols.bgp`
last_executed_command_ duration	The duration of the last executed command in seconds	A positive integer including `0`
last_executed_command_status	The status of the last executed command	• `0`: if successful • `1`: if unsuccessful
mode	The CLI mode	• `candidate`: in a configuration mode • `running`: in the running configuration • `show`: in show • `state`: in state
mode_and_session	`{mode} {configuration_ session_type}` and the candidate name	A string
mode_and_session_type	`{mode} {configuration_ session_type}`	A string
modified	`{banner}{startup_config_ state}{modified_flags}`	A string
modified_flags	The flags that indicate if the configuration is modified	• `*`: the candidate has uncommited changes • `+`: the running configuration has unsaved changed • `!`: the baseline configuration is not synchronized with the running configuration

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Keyword	Definition	Values
modified_with_change_count	**{banner}{startup_config_ state}{modified_flags}** and the count of changed objects in the candidate	• A string followed by an integer: there are changed objects in the candidate • A string: there are no changed objects in the candidate
pwc	The present working context in CLI command format	A string: for example, `network-instance default protocols bgp`
pwd	The present working directory	A string: for example, `/home/srlinux`
redundancy	The CPM redundancy status	• (a space): the CPM is the active CPM • `standby`: the CPM is the standby CPM
short_pwc	The present working context in CLI command format separated by greater than (**>**) and without key names	A string: for example, `network-instance>protocols>bgp`
short_redundancy	The short CPM redundancy status	• No value: the CPM is the active CPM • `s`: the CPM is the standby CPM
short_yang_models	The YANG models used in the CLI	• No value: the SR Linux models are used • `oc` : the OpenConfig models are used
startup_config_state	The state of the configuration loaded and committed at startup	• No value: the startup configuration succeeded • `[FACTORY]` : the startup and rescue configurations failed, and the factory default configuration succeeds • `[FAILED]` : no configuration succeeds • `[RESCUE]` : the startup configuration failed and the rescue configuration succeeds
time_12	The day of the week and time in 12-hour format	A string: for example, `Fri 1:23PM`

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Keyword	Definition	Values
time_24	The day of the week and time in 24-hour format	A string: for example, `Fri 13:23`
user	The username ($USER) in the session	A string: for example, `srlinux`
vi_editing_mode	The vi editing mode, when enabled	• `INSERT`: insert mode • `NAVIGATION`: navigation mode • `REPLACE`: replace multiple characters • `REPLACE_SINGLE`: replace a single character
xpath_pwc	The present working context in XPath format	A string: for example, `/network-instance[name=default]/protocols/bgp`

The SR Linux CLI is written using the Python Prompt Toolkit and can be customized using the styling and Python function output that are described in the toolkit documentation. In the following example, the prompt is configured to display color coded dynamic prompts and Python functions are called to display system information and the 24-hour time in HH:MM:SS format in the bottom toolbar.

```
environment prompt "\n{aaa_user}@{host} in {{ {modified_flags}{short_yang_models}{mode_and_
session} }} at [ {pwc} ]|{'<gray>took ' + last_executed_command_duration.value + 's<>' if
 last_executed_command_duration.value != '0' else ''}\n{'<green>' if last_executed_command_
status.value == '0' else '<red>'}{'#' if mode.value == 'candidate' else '$' if mode.value ==
 'state' else '❯'} "
```

```
environment bottom-toolbar "CPM:{hw_slot} {short_redundancy} {banner}{commit_confirmed_with_
remaining_time}|[{os.popen('cat /proc/loadavg').read().rstrip()}] {time.strftime('%X')}"
```

For example:

```
admin@srl1 in { candidate shared default } at [ system ]
#
CPM:A                                                [0.12 0.12 0.15 1/1267 22058] 20:44:49
```

3.5.16 Configuring default commit and discard options

About this task

You can configure the default commit operation and the default discard operation with the **environment commit-options default-operation** and **environment discard-options default-operation** commands, respectively. Default operations enable you to require specific options or set defaults that requires less typing, similar to aliases.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: Configuring commit and discard options

You can configure default commit options with the following commands:

- **environment commit-options confirmed true** – requires a commit confirmed
- **environment commit-options default-operation** – sets the default commit operation

The **environment commit-options default-operation** command has the following configuration options:

- **none** – no default operation is set, requiring an option to be entered
- **now** – sets the default commit operation to **commit now**
- **stay** – sets the default commit operation to **commit stay**

You can configure default discard options with the **environment discard-options default-operation** command.

This command has the following configuration options:

- **none** – no default operation is set, requiring an option to be entered
- **now** – sets the default discard operation to **discard now**
- **stay** – sets the default discard operation to **discard stay**

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

4 YANG data models

SR Linux supports YANG data models for configuring the network element. YANG is a standards-based, data-modelling language defined in several IETF RFCs. The SR Linux model-driven management interfaces are based on a common infrastructure that uses YANG models as the core definition for network element configuration, state, and operational actions. The model-driven interfaces (SR Linux CLI, gRPC server) take the underlying YANG modules and render them for the particular management interface.

SR Linux supports the following YANG data models:

- Nokia vendor-specific data models
- OpenConfig vendor-neutral data models

This chapter describes how SR Linux implements OpenConfig YANG data models and manages interactions with the Nokia YANG data models.

4.1 SR Linux data models

SR Linux makes extensive use of structured YANG data models to provide network operators with a single set of data models to configure and manage commonly-used network protocols, services, and devices. Each application that SR Linux supports has a Nokia vendor-specific YANG model that defines the application's configuration and state.

SR Linux exposes the YANG models to supported management APIs; for example, the CLI command tree is derived from the SR Linux YANG models loaded into the system. A gNMI client can use `Set` RPCs to configure an application based on the YANG model.

When you commit a configuration, the SR Linux management server validates the YANG models and translates them into protocol buffers for the impart database (IDB).

4.2 OpenConfig data models

OpenConfig is an informal working group that provides structured, vendor-neutral YANG data models to address the use requirements of networking applications and technologies by the community. OpenConfig data models support configuration and management of multivendor networks. They use standards-based, YANG data-modeling language, support Remote Procedure Calls (RPCs), and allow network operators to use a single set of data models to configure and manage commonly-used network protocols, services, and devices that support the OpenConfig initiative.

4.2.1 Implementation in SR Linux

SR Linux supports OpenConfig YANG data models for configuring and managing network elements. You can use the OpenConfig data models together with the SR Linux data models to configure network elements, using a CLI console or SSH connection or management-interface RPCs (gNMI) for communications between the clients and routers.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

The SR Linux and OpenConfig data models are implemented through the management server binary. The SR Linux installation process installs the data models by default during the management server installation. The OpenConfig instance of the management server passes through authorization toward the native management server.

Note:
The system does not support multiple candidates within the OpenConfig instance of the management server.

See the *SR Linux Software Release Notes* for the supported OpenConfig modules in SR Linux releases.

4.2.2 Interaction with SR Linux data models

You can use the SR Linux vendor-specific and OpenConfig vendor-neutral YANG data models together to configure and manage network elements. The SR Linux data models offer a more complete representation of the capabilities of the SR Linux network elements, because they include vendor-specific features and functions that the OpenConfig data models do not describe.

The SR Linux configuration and operational statements map to path statements in the supported OpenConfig modules. The mappings are exposed in JSON files delivered with the software package. You can view the mapping files after installation, for vendor-specific deviations and augmentations.

Note: See the *SR Linux Software Release Notes* for information about the supported OpenConfig modules in SR Linux releases.

When using the gNMI service, the capabilities RPC can discover the capabilities of a specific gNMI server. The client sends a CapabilityRequest message to request capability information from the target. The target replies with a CapabilityResponse message that includes the gNMI service version, the supported data model versions, and the supported data encodings. Subsequent RPC messages from the client use this information to indicate the set of models used by the client and the encoding used for data.

The CapabilityResponse messages indicates each data model the target supports, based on the configuration in CLI (**system management openconfig admin-state**) . The advertised names include information about the model, organization, and version, for the respective YANG models, and Nokia deviations for the OpenConfig models. See gNMI Capabilities RPC for more information.

4.2.2.1 Viewing OpenConfig to SR Linux mapping files

Procedure

SR Linux exposes the OpenConfig to SR Linux data model module mappings in `oc-srl.json` files provided with the installation. You can locate and view the module mapping files after SR Linux installation in the `/opt/srlinux/mappings/openconfig` directory.

See the *SR Linux Software Release Notes* for information about the supported OpenConfig modules in SR Linux releases.

Example:

The following example shows partial content from the `oc-srl-lldp.json` mapping file.

```
"mapping": [
  {
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
      "oc": "/lldp",
      "srlinux": "/system/srl_nokia-lldp:lldp"
    },
    {
      "oc": "/lldp/{config,state}",
      "srlinux": ""
    },
    {
      "oc": "/lldp/{config,state}/enabled",
      "srlinux": "/system/srl_nokia-lldp:lldp/srl_nokia-lldp:admin-state",
      "transform":"bool-to-admin-state"
    },
    {
      "oc": "/lldp/{config,state}/hello-timer",
      "srlinux": "/system/srl_nokia-lldp:lldp/srl_nokia-lldp:hello-timer"
    },
    {
      "oc": "/lldp/{config,state}/suppress-tlv-advertisement",
      "supported": false
    },
    {
      "oc": "/lldp/config/system-name",
      "supported": false
    },
    {
      "oc": "/lldp/state/system-name",
      "srlinux": "/system/srl_nokia-lldp:lldp/srl_nokia-lldp:system-name"
    },
    {
      "oc": "/lldp/config/system-description",
      "supported": false
    },
    {
      "oc": "/lldp/state/system-description",
      "srlinux": "/system/srl_nokia-lldp:lldp/srl_nokia-lldp:system-description"
    },
    {
      "oc": "/lldp/config/chassis-id",
      "supported": false
    },
    {
      "oc": "/lldp/state/chassis-id",
      "srlinux": "/system/srl_nokia-lldp:lldp/srl_nokia-lldp:chassis-id"
    },
    {
      "oc": "/lldp/config/chassis-id-type",
      "supported": false
    },
    {
      "oc": "/lldp/state/chassis-id-type",
      "srlinux": "/system/srl_nokia-lldp:lldp/srl_nokia-lldp:chassis-id-type"
    },
    {
      "oc": "/lldp/state/counters",
      "srlinux": "/system/srl_nokia-lldp:lldp/srl_nokia-lldp:statistics"
    },
    {
      "oc": "/lldp/state/counters/frame-in",
      "srlinux": "/system/srl_nokia-lldp:lldp/srl_nokia-lldp:statistics/srl_nokia-
lldp:frame-in"
    },
    {
#
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

4.2.3 Viewing the nosimage OpenConfig capability file

Introduction

SR Linux supports a machine-readable description of OpenConfig YANG paths and the gNMI and gRPC operations that apply to each path. The description is delivered as a textual `protobuf` file called `nosimage.textproto`. You can use this file offline to design automation, validate client requests, and plan telemetry subscriptions without connecting to a router.

This topic describes how to obtain `nosimage.textproto`, how paths are represented, and how to read gNMI `get`, `set`, and `subscribe` capabilities, subscription modes, stream modes, and supported OpenConfig gRPC services.

About `nosimage.textproto`

`nosimage.textproto` is a human-readable instance of data defined by the `nosimage.proto` schema. The file references `ocpaths.proto` for OpenConfig path entries and `ocrpcs.proto` for supported gRPC services. Each release builds the file from the OpenConfig YANG models supported on SR Linux, including `openconfig-srl-augments.yang` and `openconfig-srl-deviations.yang`.

The file lists OpenConfig paths only. It does not describe Nokia native (srl_nokia) YANG. YANG action paths and action or RPC input and output containers are not included.

Note: `nosimage.textproto` documents capabilities for a release and platform. It does not change router behavior. Runtime access still requires OpenConfig to be enabled and appropriate gNMI or gRPC configuration on the device.

Platforms and delivery

Nokia provides a platform-specific `nosimage.textproto` file with the SR Linux software delivery for each supported chassis type. Per-platform files are available for the following platform variants.

- 7250 IXR-10e
- 7250 IXR-18e

Download the SR Linux release package for your platform from the Nokia software delivery portal and locate `nosimage.textproto` with the other release deliverables. Content can differ between platforms because supported OpenConfig paths depend on platform capabilities.

OpenConfig path format

Each path in `nosimage.textproto` is an absolute path from the OpenConfig data tree root. Paths are written without list key values. For example, a list may appear as `/interfaces/interface/state/oper-status` and not including a specific interface name.

The path list includes every supported OpenConfig node where gNMI `get`, `set`, or `subscribe` can apply. Container and list paths that support subscription are included when applicable.

gNMI capabilities per path

Path entries can include a `gnmi_rpc` block that indicates which gNMI RPCs are supported for that path. The following fields are used.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Table 9: gnmi_rpc fields per OpenConfig path

Field	Meaning
`get`	The path can be read with the gNMI Get RPC. All YANG paths exce support Get, including both config true and config false nodes.
`set`	The path can be written with the gNMI Set RPC. Set is true for YAN
`subscribe`	The path can be used with the gNMI Subscribe RPC when the Oper the node with the telemetry-capability extension (sample, on_chang

Subscription modes

When `subscribe` is `true`, `nosimage.textproto` lists one or more `sub_mode` values that describe how the target can deliver updates for that path, as defined in the gNMI specification:

- **STREAM** — the target streams values for the subscription.
- **ONCE** — the target sends values once and closes the subscription.
- **POLL** — the target sends updates when the client sends a poll request.

On SR Linux, Subscribe-capable paths typically list `STREAM`, `ONCE`, and `POLL` together.

Stream modes and sample interval

When `sub_mode` includes `STREAM`, `stream_mode` indicates how streamed updates are produced for that path. Values are derived from the telemetry-capability extension on the YANG node:

Table 10: Telemetry-capability to stream_mode mapping

Telemetry-capability in YANG	`stream_mode` in `nosimage.textproto`
`target_defined`	`TARGET_DEFINED`
`on_change`	`ON_CHANGE`
`sample=<n>`	`SAMPLE`

When `stream_mode` includes `SAMPLE`, `sample_interval_nanoseconds` gives the interval associated with sample=<n> in YANG. The extension uses seconds; `nosimage.textproto` stores nanoseconds (seconds × 10^9).

Supported gRPC services

The `ocrpcs` section of `nosimage.textproto` lists OpenConfig-related gRPC services, supported method names, and service version strings. Use this section together with `per-path gnmi_rpc` data when you implement client capability checks.

Example `ocrpcs` structure is as follows.

```
ocrpcs: {
  oc_protocols: {
    key: "gnmi"
    value: {
      method_name: "gnmi.gNMI.Get"
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
      method_name: "gnmi.gNMI.Set"
      method_name: "gnmi.gNMI.Subscribe"
      version: "0.10.0"
    }
  }
}
```

Method names use fully qualified gRPC identifiers (for example, gnoi.healthz.Healthz.Get).

Example path entry

The following example shows a path entry with gNMI capability and subscription information. Path and field values are illustrative.

```
ocpaths: {
  path: "/interfaces/interface/state/oper-status"
  gnmi_rpc: {
    get: true
    set: false
    subscribe: true
    sub_mode: STREAM
    sub_mode: ONCE
    sub_mode: POLL
    stream_mode: ON_CHANGE
  }
}
```

Using nosimage with management on the router

`nosimage.textproto` complements but does not replace on-device discovery and mapping artifacts.

- Enable OpenConfig with `system management openconfig admin-state enable` before using OpenConfig paths through gNMI.
- Set the gNMI origin to openconfig in request path prefixes, or configure `system grpc-server yang-models openconfig` on the gRPC server.
- Use the gNMI Capabilities RPC on a live device to learn gNMI service version, supported YANG models, and encodings. Capabilities does not provide per-path Subscribe or stream mode detail; use `nosimage.textproto` for that level of detail.
- After installation, OpenConfig-to-native path mappings remain in `/opt/srlinux/mappings/openconfig` as `oc-srl.json` files.

4.2.4 Enabling access to OpenConfig data models

Procedure

You can enable access to the OpenConfig data model using the **admin-state** in the **system management openconfig admin-state** context. When you validate and commit a configuration, the system verifies if **openconfig** is set to **true** and determines if there is access to OpenConfig data models. If the candidate contains OpenConfig configuration statements and **openconfig** is not enabled (**false**), the action fails with an error.

Note: The system does not support multiple candidates within the OpenConfig instance of the management server.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

The following example shows the basic configuration required to enable the OpenConfig configuration modules and obtain state information.

Example:

```
--{ [FACTORY] candidate shared default }--[ ]--
A:dut2# system management openconfig admin-state enable
--{ [FACTORY] * candidate shared default }--[ ]--
A:dut2# commit stay
All changes have been committed. Starting new transaction.
--{ [FACTORY] + candidate shared default }--[ ]--
A:dut2#
```

4.2.5 Specifying the data model mode

The SR Linux YANG data model is the default mode. The system supports different methods for switching between the default SR Linux data model and the OpenConfig data model.

4.2.5.1 Specifying the data model in the CLI

Procedure

To specify the data model when accessing the CLI, use the **enter oc** or **enter srl** command. A special OpenConfig banner appears in OpenConfig mode.

Example: Specifying the data model

```
A:dut2# enter oc
--{ oc candidate shared default }--[ ]--
A:dut2# enter srl
--{ [FACTORY] + candidate shared default }--[ ]--
A:dut2#
```

4.2.5.2 Specifying the data model in Linux Bash

Procedure

To specify the data model when logging in to the Linux Bash, start the CLI using **sr_cli --oc** or **sr_cli --srl**. This enters directly into OpenConfig or SR Linux models respectively.

Example: Specifying the data model in Linux Bash

```
$ sr_cli --oc
Loading environment configuration file(s): []
Welcome to the Nokia SR Linux CLI.

--{ oc running }--[ ]--
A:srl1# quit
Connection to srl1 closed.
$ sr_cli --srl

Loading environment configuration file(s): []
Welcome to the Nokia SR Linux CLI.
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
--{ [FACTORY] + running }--[ ]--
A:srl1# quit Connection to srl1 closed.
```

4.2.5.3 Specifying the data model mode for the gRPC server

Procedure

When using the gRPC server to send and accept edits, requests, and responses for the SR Linux or OpenConfig data models, you can specify the corresponding data model schema (**openconfig** or **native**) for the target in the **system grpc-server** *name* **yang-models** context. The following example shows this configuration.

Example: Yang data model selection for gRPC server

```
A:dut2# enter candidate
--{ [FACTORY] + candidate shared default }--[ ]--
A:dut2# system grpc-server mgmt yang-models openconfig
--{ [FACTORY] +* candidate shared default }--[ ]--
A:dut2# commit stay
All changes have been committed. Starting new transaction.
--{ [FACTORY] + candidate shared default }--[ ]--
A:dut2#
```

Alternatively, you can send a gNMI SetRequest with the origin prefix **openconfig** embedded in the request. This informs the system to use the OpenConfig data model, regardless of the value specified for **yang-models** under the gRPC server configuration.

Example: Prefix origin openconfig in gNMI SetRequest

```
SetRequest:
prefix: <
  origin: "openconfig"
>
```

4.2.6 Verifying the OpenConfig operational state

Procedure

You can verify the OpenConfig operational state using the command **info from state system management openconfig oper-state**. The following example shows the operational state of the OpenConfig data model.

Example:

```
A:dut2# info with-context from state system management openconfig
system {
    management {
        openconfig {
            admin-state enable
            oper-state up
        }
    }
}
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

5 YANG-modeled operations

The SR Linux CLI supports YANG-modeled operational commands in the application data model.

SR Linux defines these commands as YANG action statements as defined in RFC7950. An action executes an operation on the device, such as clearing statistics or resetting protocol information. Actions typically return structured output, although some actions may display unstructured text when executed from the CLI. The model defines each action in the application context where it applies. An action that clears BGP malformed-update information, for example, belongs to the BGP context of a network instance.

You can execute an action using the `tools` command. The `tools` command is global and can be entered from any context. The **tools** command must be used to explicitly enable the action's execution. This prevents actions from being running accidentally when navigating through the CLI tree.

5.1 YANG model of an action

An action is defined using the `action` statement specified in RFC 7950. The action is attached to the operational hierarchy of the feature it operates on, allowing the action to be associated with the relevant object or resource. For example, the `restart` action in the application management hierarchy is attached to a specific application instance. The invocation path identifies the target application through the list key (`name`), so the application name does not need to be repeated in the action input.

The `input` and `output` statements are optional. The `input` statement defines parameters provided when invoking the action, while the output statement defines the data returned by the action. Input parameters are typically modeled as leaves or leaf-lists. If the operation does not require additional parameters, the `input` statement can be omitted. Similarly, the `output` statement can be omitted when indicating success or failure is sufficient.

The following example shows a `restart` action attached to an operational application list entry.

```
module: srl_nokia-app-mgmt
  augment /srl-sys:system:
    +--ro app-management
       +--ro application* [name]
          +--ro name                    string
          +---x restart {tools-as-modeled-actions-different-io-app-mgr}?
             +---w input
             |  +--w (type-of-restart)?
             |     +--:(warm-case)
             |     |  +---w warm          empty
             |     +--:(cold-case)
             |        +---w cold          empty
             +--ro output
                +--ro sent-signal         uint32
                +--ro sent-signal-name    string
                +--ro delayed            boolean
```

The preceding example illustrates an action with both `input` and `output` parameters. The `input` allows the caller to request either a warm or cold restart of the specified application. The `output` reports information about the signal sent to the application and whether the restart operation was delayed. The

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

target application is identified by the application list key in the invocation path, so the application name is not included in the action input.

Unstructured output

Some YANG modeled actions in SR Linux may optionally provide unstructured data as an alternative to the prefered structured data.

SR Linux YANG model that provide this unstructured-data is modeled using an SR Linux reserved key-less YANG list named `srl-unstructured-output`. This field includes formatted, human readable, unstructured data only and it is therefore not the prefered approach when automating any activity on SR Linux.

Data provided in the `srl-unstructured-output` YANG list is returned on the CLI only, and only when either:

- The operator explicitly requests this output format. In this situation, only the contents of the `srl-unstructured-output` list is displayed and none of the structured data modelled in that actions output section is returned.
- There is no structured output and the operator has configured the `tools-unstructured-fallback` option.

5.2 Executing an action

Execute an action from the context where it is defined, or from anywhere else by entering the full path as shown in the following example.

Example: Execute an action

```
--{ + candidate shared default }--[  ]--
# tools network-instance default protocols bgp clear-malformed-updates
```

```
--{ + candidate shared default }--[  ]--
# tools network-instance default protocols bgp neighbor 10.2.2.1 clear-malformed-updates
```

```
--{ + candidate shared default }--[ network-instance default protocols bgp ]--
# tools clear-malformed-updates
```

```
--{ + candidate shared default }--[ system grpc-server mgmt ]--
# tools /network-instance default protocols bgp clear-malformed-updates
```

5.3 Finding an action

Enter the ? command to list the commands and actions in the current context. The CLI lists configuration commands under local commands and lists actions under local actions. To list only the actions, enter `tools` and press Tab.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: Local commands and local actions

```
--{ running }--[ network-instance default protocols bgp neighbor fd00:fde8::2:13 ]--
# ?
Local commands:
  admin-state*      Administratively enable or disable the peer
  afi-safi          List of address families supported by the BGP neighbor
  as-path-options   Options for handling the AS_PATH in received BGP routes
  authentication    Container with authentication options that apply to this specific peer
  description*      A user provided description string for the peer
  export-policy*    Apply an export policy to advertised BGP routes
  failure-detection
                    Options related to methods of detecting BGP session failure
  graceful-restart  Options related to router behavior as a graceful restart helper
  import-policy*    Apply an import policy to received BGP routes
  local-as          Options related to the local autonomous-system number advertised by
 this router to the peer
  local-preference*
                    The value of the local-preference attribute that is added to received
 routes from the peer, if it is EBGP
  multihop          Configuration parameters specifying the multihop behaviour for an EBGP
 peer. This is not applicable to an IBGP peer.
  next-hop-self*    When set to true, the next-hop in all IPv4-unicast, IPv6-unicast and
 EVPN BGP routes advertised to the peer, if IBGP, is set equal to the local-address used
 on this session (or to the router ID if the NLRI is IPv6 and there
                     is no IPv6 local address to use). This is independent of the route
 origin (EBGP, IBGP-client, IBGP-non-client or redistributed direct/static/aggregate
 route).
  peer-as*          The autonomous system number expected from the peer
  peer-group*       A reference to the peer-group template to use for this BGP session
  route-flap-damping*
                    Enable/disable route flap damping procedures for routes received from
 this peer if it is an EBGP peer
  route-reflector   Container with route reflection configuration options.
  send-default-route
                    Options for controlling the generation of default routes towards the
 peer
  timers
  trace-options     Debug traceoptions for BGP
  transport

Local actions:
  soft-clear        Perform a soft reset of the BGP neighbor
  reset-peer        Perform a hard reset of the BGP neighbor

*** Not all commands are listed, press '?' again to see all options ***
```

Example: Press tab after tools to list the local actions only

```
--{ + running }--[ network-instance default protocols bgp neighbor 10.2.2.2 ]--
# tools
                    ..                          reset-peer
                    /                           route-flap-damping
                    clear-malformed-updates     soft-clear
```

Example: `tree from tools` command

Use the `tree from tools` command to display the actions in the current context. Add a path to display the actions under that path. Add `detail` or `extensive` to show the input and output of each action. Add `flat` to display each node on one line. You can combine `flat` with `detail` or `extensive`.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- ? indicates an optional node.
- ^I indicates input to the action.
- ^0 indicates output from the action.
- x indicates the action itself. `tree from tools extensive` shows this flag.

```
--{ + running }--[  ]--
# tree from tools / network-instance protocols bgp neighbor
neighbor
+-- reset-peer
+-- soft-clear
|  +-- route-refresh
+-- route-flap-damping
|  +-- clear-history
+-- clear-malformed-updates
```

Example: `tree from tools detail` command

```
--{ + running }--[  ]--
# tree from tools detail / network-instance protocols bgp neighbor
neighbor* [peer-address]    bgp_mgr
+-- reset-peer?    bgp_mgr
+-- soft-clear!    bgp_mgr
|  +-- route-refresh?    bgp_mgr
+-- route-flap-damping!    bgp_mgr
|  +-- clear-history?    bgp_mgr
+-- clear-malformed-updates
```

Example: `tree from tools extensive` command

```
--{ + running }--[  ]--
# tree from tools extensive / network-instance protocols bgp neighbor
rw neighbor* [peer-address]    bgp_mgr
+--rw peer-address  srl_nokia-common:ip-address-with-zone    bgp_mgr
+--rw reset-peer?  empty    bgp_mgr
+--rw soft-clear!    bgp_mgr
|  +--rw route-refresh?  identityref    bgp_mgr
+--rw route-flap-damping!    bgp_mgr
|  +--rw clear-history?  empty    bgp_mgr
+--x  clear-malformed-updates
```

Example: `tree from tools flat` command

```
--{ + running }--[  ]--
# tree from tools flat / network-instance protocols bgp neighbor
network-instance protocols bgp neighbor reset-peer
network-instance protocols bgp neighbor soft-clear route-refresh
network-instance protocols bgp neighbor route-flap-damping clear-history
network-instance protocols bgp neighbor clear-malformed-updates
```

Example: `tree from tools flat detail` command

```
--{ + running }--[  ]--
# tree from tools flat detail / network-instance protocols bgp neighbor
network-instance* [name] protocols bgp!+ neighbor* [peer-address] reset-peer?    bgp_mgr
network-instance* [name] protocols bgp!+ neighbor* [peer-address] soft-clear! route-
refresh?    bgp_mgr
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
network-instance* [name] protocols bgp!+ neighbor* [peer-address] route-flap-damping!
 clear-history?    bgp_mgr
network-instance* [name] protocols bgp!+ neighbor* [peer-address] clear-malformed-updates
network-instance* [name] protocols bgp!+ neighbor* [peer-address] clear-malformed-updates
```

Example: `tree from tools flat extensive` command

```
--{ + running }--[  ]--
# tree from tools flat extensive / network-instance protocols bgp neighbor
rw network-instance* [name] protocols bgp!+ neighbor* [peer-address] peer-address  srl_
nokia-common:ip-address-with-zone    bgp_mgr
rw network-instance* [name] protocols bgp!+ neighbor* [peer-address] reset-peer?  empty
 bgp_mgr
rw network-instance* [name] protocols bgp!+ neighbor* [peer-address] soft-clear! route-
refresh?  identityref    bgp_mgr
rw network-instance* [name] protocols bgp!+ neighbor* [peer-address] route-flap-damping!
 clear-history?  empty    bgp_mgr
x  network-instance* [name] protocols bgp!+ neighbor* [peer-address] clear-malformed-
updates
ro network-instance* [name] protocols bgp!+ neighbor* [peer-address] clear-malformed-
updates
```

5.4 Displaying action results

The CLI displays action output as the action produces it. The command completes when the action completes. If the action fails, the CLI displays a message, even if it has already displayed some output. The device also records any failures in the system log. When an action succeeds and returns no data, the CLI displays no output.

5.5 Formatting action output

Use the `as` output modifier to choose the format of one action command. The modifier overrides the default format for that command as described in the following sections.

Example: Display `as text`

The `as text` command displays the structured output in CLI text. This is the default option.

```
--{ + running }--[  ]--
# tools system ipfix flow-cache dump-flows ipv4 | as text
    flow-dump "type: 1

The top 20 IPv4 unicast flows seen by cflowd are:

     Current Time: 10/14/2026 12:57:33
Last Cleared Time: 10/14/2026 12:49:01
   ifIndexContext: global

Intf/Ingr  SrcIP             Intf/Egr  DstIP                          Pro     ToS
I-vRtrID   S-Port Msk AS     E-vRtrID  D-Port Msk AS                 Flgs    Pkts
                                       NextHop                  Pkt-Size    Time
-------------------------------------------------------------------------------
98304      130.210.240.1     0         130.195.80.2                     1    0x00
10242      0      /32 0      0         0      /24 0                 0x00     338
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
                                    0.0.0.0                       84      68
98304      130.210.240.1     0      130.195.80.2                   1    0x00
10242      0      /32 0      0      0      /24 0                0x00     335
                                    0.0.0.0                       84      68
98304      130.210.240.1     0      130.195.80.2                   1    0x00
10242      0      /32 0      0      0      /24 0                0x00     335
                                    0.0.0.0                       84      68
98304      130.210.240.1     0      130.195.80.2                   1    0x00
10242      0      /32 0      0      0      /24 0                0x00     330
                                    0.0.0.0                       84      67
98304      130.210.240.1     0      130.210.240.2                  1    0x00
10242      0      /32 0      0      0      /32 0                0x00       3
                                    0.0.0.0                       84       2
-------------------------------------------------------------------------------

"
```

Unstructured output

The `as unstructured-output` command displays the unstructured data output, if defined in the special srl-unstructured-output YANG list of the action. If the action has structured output, it is not displayed. If the action has no unstructured output, the CLI displays nothing. See Unstructured output for more information.

The `environment tools-unstructured-fallback` configuration can be set to `true` to allow the CLI to display unstructured output if the action does not define any structured output. Fallback is enabled by default.

```
--{ + running }--[  ]--
# environment tools-unstructured-fallback true
```

Use the `environment tools-output-format` command to set the default display format for `tools` commands that do not include an `as` modifier. Valid values are `text`, `json`, `yaml`, `xml`, and `unstructured-output`. The default is `text`.

With `text`, `json`, `yaml`, or `xml`, the CLI renders the action's structured output in that format and omits unstructured output. With `unstructured-output`, the CLI displays only the unstructured lines defined for the action.

The **environment tools-output-format** command sets the default encoding for YANG-modeled tools actions; The **environment output-format** command sets the default for **info**, **show**, and other non-tools CLI output. The two settings are independent and appear separately in environment **show** and **srlinux.rc**.

```
--{ + candidate shared default }--[  ]--
# system cli environment tools-output-format?
usage: tools-output-format <json|text|unstructured-output|xml|yaml>

The CLI output display format applied to 'tools' commands. Set to 'text' by default

Selecting 'text', 'json', 'yaml' or 'xml' causes the action's
structured 'output' updates to be rendered using the standard data
formatter. Selecting 'unstructured-output' instead displays only the
action's free-form 'standard_output' channel.

Positional arguments:
  value             [json|text|unstructured-output|xml|yaml, default text]
```

The tools output format is stored in the global `srlinux.rc` file.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
[tools-output-format]
# The default CLI output display format applied to 'tools' (YANG-modeled
# action) commands. It is used when the user does not specify the output
# modifier (e.g. '| as json') in the tools command itself.
# Selecting 'text' (the default), 'json', 'yaml' or 'xml' renders the action's
# structured 'output' updates through the standard data formatter and
# suppresses the unstructured channel; selecting 'unstructured-output' displays
# only the action's free-form 'standard_output' channel.
# Note: 'table' is intentionally not available for 'tools' commands because
# action outputs are not tabular. Use [output-display-format] for non-action
# commands.
# Supported values are: "text", "json", "yaml", "xml", "unstructured-output"
#
# Default value:
# value = "text"
```

usage: environment

The `environment` command lists the tools settings with the other environment commands.

```
--{ running }--[  ]--
# environment
usage: environment

Control the look-and-feel of the CLI

Local commands:
  alias             Create or overwrite an alias
  basic-prompt      Change the prompt for 'basic' cli-engine type displayed before every input line
  bottom-toolbar    Change the text displayed in the bottom toolbar.
  cli-engine        CLI engine settings
  commit-options    Commit options settings
  complete-on-enter
                    Trigger auto-completion when you press Enter
  complete-on-space
                    Trigger auto-completion whenever you press Spacebar
  complete-on-tab   Trigger auto-completion whenever you press Tab
  delete            Reset/remove environment settings
  discard-options   Discard options settings
  key-completer-limit
                    The limit on number of keys shown in auto-completion
  load              Load the environment settings from a file
  network-instance  The default network-instance to be used if not specified
  output-format     The CLI output display format. Set to 'text' by default.
  output-modifier-alias
                    Create or overwrite an output modifier alias
  pagination        Pagination override of the CLI output, either "on" or "off"
  prompt            Change the prompt for 'advanced' cli-engine type displayed before every input line
  save              Store the environment settings to a file
  session-idle-timeout
                    The idle timeout override for the CLI session in seconds
  show              Show the currently active environment settings
  tools-output-format
                    The CLI output display format applied to 'tools' commands. Set to 'text' by
 default.
  tools-unstructured-fallback
                    Fall back to a 'tools' action's free-form standard_output when the
  yang-models       The default yang-models to be used when starting the CLI, either "srl" or "oc"

*** Not all commands are listed, press '?' again to see all options ***
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

6 RPC overview and supporting interfaces

SR Linux supports the gNMI and JSON interfaces that use the Remote Procedure Call (RPC) protocol for the modification and retrieval of a configuration from a target device as if it were a local object. This chapter provides a general RPC overview and defines how SR Linux implements the gNMI and JSON serving RPCs.

6.1 RPC overview

An RPC executes a procedure or method on a remote device in a way similar to one executed locally. Using an RPC should look and feel like a local procedure call and achieve similar results.

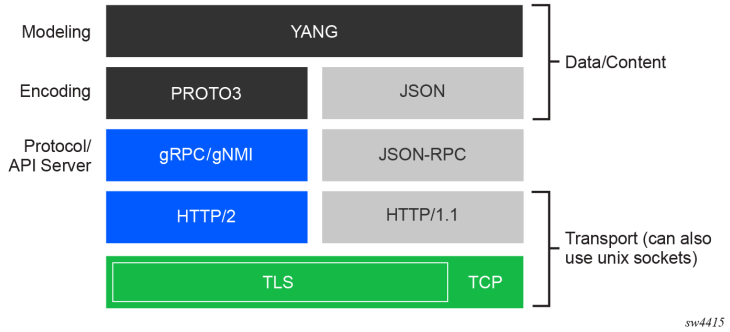
When RPCs operate in a server/client model, the client executes an application and requests some method be executed on a server. The server receives the RPC, executes the method, and sends the outcome back to the server. For this to occur, the following are needed:

- A server application that contains a set of methods (or service) which a client can call. For example, a gRPC server or JSON-RPC server. Each needs to receive an RPC from a client for a specific method or service, execute the requested methods/service, and return data.
- A common way of describing which method to execute and associated data.

  This is referred to as encoding/decoding or serialization/deserialization. For example, proto buffers or JSON.
- A transport mechanism between the two devices. In the case of gNMI and JSON-RPC this is done using HTTP/2 and HTTP1.1 over TCP secured by TLS.

For RPCs to relate to network applications and a network operating system, there needs to be a way to define the data model of the network constructs which an RPC must perform. In the SR Linux, all applications are modeled in YANG, as shown in the following figure.

Figure 8: RPC-based interfaces in SR Linux



© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

6.1.1 gNMI path convention

Because SR Linux is modeled in YANG, for RPCs to retrieve or configure an SR Linux device (set) the data location or path to the data within the YANG model must be specified to the RPC. For example, to configure a description under a BGP peer, the RPC server must be able to reference this specific data and its location in the SR Linux data model.

Both the JSON-RPC and gNMI use the gNMI path convention to describe the location or path in SR Linux. The gNMI path convention is commonly referred to as the "path" and defines the data structure of a path to reference a config or state leaf within SR Linux.

A path is an ordered list of path elements with each element containing an element name and one or more key value pairs associated with the element. For example:

```
path: <
  elem: <
    name: "interface"
    key: <
      key: "name"
      value: "ethernet-1/20"
    >
  >
  elem: <
    name: "subinterface"
    key: <
      key: "index"
      value: "1"
    >
```

A path can also be displayed as a string which is more human readable. The rules for building this string type are:

- Each path element is separated by a "/" character and listed in sequence.
- A path element which contains one or more key/value pairs is represented by the path element name followed the key/value wrapped in brackets ([ ]).
- The root of the data tree is represented by a single "/".

The following is a human readable path for the previous example:

```
/interface=ethernet-1/20/subinterface[index=1]
```

6.2 Configuring a gRPC or JSON server

SR Linux can enable a gRPC server that allows external gRPC clients to connect to the device and modify the configuration and collect state information. You can also enable a JSON-RPC server on the SR Linux device, which allows you issue JSON-formatted requests to the device to retrieve and set configuration and state.

See the Management Servers chapter in the *SR Linux Configuration Basics Guide* for details on how to configure these servers.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

7 gNMI

gRPC Network Management Interface (gNMI) is a gRPC based protocol that defines a service or set of services or RPC methods used to configure and retrieve data from network devices.

SR Linux provides a gNMI-based RPC for the modification and retrieval of a configuration. Supported RPCs are:

- Get
- Set
- Subscribe
- Capabilities

7.1 Common notification messages

When the SR Linux gNMI process communicates data to a client, it uses common notification messages. Notification messages use the fields shown in the following table.

Table 11: Common notification fields

Field	Definition
timestamp	Time data was collected
prefix	Prefix applied to all path fields included in the notification message. The paths expressed within the message are formed by the concatenation of prefix + path.
update	List of update messages that indicate changes in the underlying data. Subfields are: • path • val (value)
delete	List of paths indicating the deletion of data nodes

7.1.1 Timestamps

Timestamp values are represented in nanoseconds. The value is encoded as a signed 64-bit integer (int64).

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

7.1.2 Path prefix

A prefix can be specified to reduce the lengths of path fields within a message. The absolute path is a concatenation of the path elements representing the prefix and the list of path elements in the path field. For example:

Example: Path Prefix

```
notification: <
timestamp: (timestamp) // timestamp as int64
prefix: <
  elem: <
      name: "a"
    >
  elem: <
     name: "b"
     key: <
        key: "name"
        value: "b1"
       >
     >
  elem: <
     name: "c"
  >
>
update: <
  path: <
    elem: <
      name: "d"
    >
  >
  value: <
    val: <
      json_val: "AStringValue"
    >
  >
>
update: <
  path: <
    elem: <
      name: "e"
    >
  >
  val: <
    json_val: 10042 // converted to int representation
  >
 >
>
```

7.1.3 Paths

Paths are represented according to gNMI path conventions. Each path is represented by an ordered list of PathElem messages, starting at the root node, and ending at the most specific path element (versus a single string with a "/" character separating each element). Each PathElem message contains the name of the node within the data tree, along with any associated keys and attributes that may be required. For example:

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example

```
path: <
  elem: <
    name: "a"
  >
  elem: <
    name: "e"
    key: <
      key: "key"
      value: "k1"
    >
  >
  elem: <
    name: "f"
  >
  elem: <
    name: "g"
  >
>
```

Multiple paths are supported. Multiple notification messages are triggered in response to each path. For example:

Example

```
path <
  elem <
    name: "interface"
    key <
      key: "name"
      value: "mgmt0"
    >
  >
>
path <
  elem <
    name: "system"
  >
>
type: 1
encoding: JSON_IETF
```

7.1.4 Data node values

The value of a data node can be the following:

- scalar types, such as a string in the string_val field, int64 in the int_val field, unit64 in the uint_val field, bool in the bool_val field, bytes, and float in the float_val field
- additional types used in some schema languages, such as decimal64 in the decimal_val field and ScalarArray in the leaflist_val field
- structured data types

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

7.1.4.1 Structured data types

When structured data is sent in an update message, it is serialized according to supported encoding, as shown in the following table.

Table 12: Encoding for structured data

Data type	Description	Field
ASCII	An ASCII encoded string	ascii_val
PROTO	A Protobuf encoded message using protobuf.any	any_val
JSON_IETF	A JSON encoded string using JSON encoding compatible with RFC 7951	json_ietf_val

7.2 Data model selection using gNMI origin

Within a gNMI request, the origin extension allows clients to specify which data model to interact with: OpenConfig (**openconfig**), native (**native** or **srlinux_native**), or CLI (**cli** or **srlinux_cli**).

Clients can specify a value for the origin in the path prefix or in one or more paths, but not both at once. When a request specifies the origin in the path prefix, SR Linux uses the specified data model to process all operations within the transaction.

Note: If a request includes a path prefix, the prefix must specify any required origin.

If no origin is specified in a request, SR Linux handles the request based on the data model specified under **system grpc-server yang-models** (set to **native** by default).

SR Linux deviates from the gNMI specification in that it intrinsically links all origins such that a change in one data model produces the matching change in the other.

The following table describes the usage of the origin field in the message types where it is typically used.

Table 13: Purpose of origin field by message type

Message type	Purpose of `origin` field
SetRequest	Specifies the schema to use to modify the target configuration
GetRequest	Retrieves the contents of the specified schema
GetResponse	Indicates that the payload contains data from the specified <origin, path> schema
SubscribeRequest	Subscribes to paths within the specified schema
SubscribeResponse	Indicates the update corresponds to the specified <origin, path> tuple

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Multiple path origins

If a SetRequest specifies more than one origin (for example, it contains two operations each with different path origins), the updates are processed as a single transaction. In this case, all operations must succeed for the SetResponse to return a success message. If any of the operations fail, the contents of all origins roll back, and the SetResponse returns an error.

You can specify mixed origins combining OpenConfig with native YANG or with CLI. In either case, SR Linux applies the OpenConfig origin first, followed by the native or CLI origin.

The following example shows the OpenConfig origin defined in the path prefix:

Example: Origin defined in path prefix

```
prefix {
  origin: "openconfig"
}
path {
  elem {
    name: "interfaces"
  }
  elem {
    name: "interface"
    key {
      key: "name"
      value: "ethernet-1/1"
    }
  }
  elem {
    name: "state"
  }
  elem {
    name: "description"
  }
}
type: ALL
encoding: JSON_IETF
```

7.2.1 CLI configuration with gNMI origin

SR Linux supports setting the gNMI origin to CLI (**cli** or **srlinux_cli**), which allows gNMI clients to apply CLI configurations. In this case, SR Linux ignores any defined path and enters and commits the configuration included in the update or replace operation as CLI input.

The value defined in the CLI origin is encoded as ASCII and supports both the tree-based CLI commands as displayed using the SR Linux **info** command or the full-context **set /** commands as shown using the **info flat** command.

Delete operations are not supported with origin CLI.

If a SetRequest replace operation contains only the CLI origin, SR Linux treats the operation as a full device configuration replacement. The whole running configuration is replaced with the CLI commands applied over a blank configuration.

SR Linux supports only one CLI replace operation per SetRequest. However, one or more update operations can follow the replace operation, in which case the updates are appended to the contents of the replace operation.

The following example shows a SetRequest that uses CLI origin within the update operation:

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: SetRequest with CLI origin

```
update: <
  path: <
    origin: "cli"
  >
  val: <
    ascii_val: "/interface ethernet-1/1 admin-state disable"
  >
>
```

7.3 gNMI Get RPC

The Get RPC allows you to obtain a view of the existing state. A GetRequest message is sent to the target (SR Linux gNMI process gnmi_mgr) that specifies the data to retrieve. A GetResponse message is returned that reflects the values of specified leafs at the collection time.

The Get RPC is recommended for retrieving small data sets. For larger data sets, the gNMI subscribe RPC is recommended, using the ONCE mode.

Related topics

gNMI Subscribe RPC

7.3.1 GetRequest message

A GetRequest message retrieves a view of data from the server. A Get RPC requests the server retrieve a subset of the data tree as specified by the paths included in the message and serializes this using the specified encoding. The GetRequest message uses the fields shown in the following table.

Table 14: GetRequest fields

Field	Definition
path	Path (or set of paths) for the requested data view. Wildcards are permitted.
type	Type of data requested. Supported options are: • CONFIG (configurable read/write data) • STATE (non-configurable read-only data)
encoding	Encoding that the target should use (ASCII or JSON_IETF). If not specified, JSON is the default.
extension	Repeated field to carry gNMI extensions

7.3.2 GetResponse message

The GetResponse message uses the fields shown in the following table.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Table 15: GetResponse fields

Field	Definition
notification	Set of notification messages for each path specified in the Get Request.
extension	Repeated field to carry gNMI extensions

Related topics

Common notification messages

7.4 gNMI Set RPC

The Set RPC allows you to modify an existing state. A SetRequest message is sent to the target (SR Linux gNMI process gnmi_mgr) that specifies the required modifications. The server deletes, replaces, and updates paths based on the order they are listed. For each operation designated in the SetRequest message, an UpdateResult message is included in the SetResponse message.

7.4.1 SetRequest message

The SetRequest message uses the fields shown in the following table.

Table 16: SetRequest fields

Field	Definition
prefix	A specified prefix is applied to all defined paths within each field
delete	A set of paths to be removed from the data tree
replace	A set of update messages that defines content to replace
union_replace	A set of update messages of mixed origin that defines content to replace
update	A set of update messages that defines content to update
extension	Repeated field to carry gNMI extensions

An update message indicates changes to paths where a new value is required. Update messages contain the following:

- path - the path of the element to be modified
- value - a value to apply to the specified node

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

All changes to the state included in a SetRequest message are considered part of a transaction. Either all modifications are applied or changes are rolled back to reflect the original state. For changes to be applied together, they must be in a single SetRequest message.

For replace operations, the behavior of omitted data elements depends on whether they are non-default values (set by a previous SetRequest message) or unmodified defaults. When the replace operation omits values that have been previously set, they are deleted from the data tree. Otherwise, omitted data elements are created with their default values.

For update operations, only the value of the data elements explicitly specified are changed.

7.4.2 SetResponse message

The SetResponse message uses the fields shown in the following table.

Table 17: SetResponse fields

Field	Definition
prefix	The prefix specified for all paths
response	A list of responses (one per operation). Each response consists of an UpdateResult message with the following: • timestamp - time when the SetRequest message was accepted • path - path defined in SetRequest message. A prefix may be present to reduce repetition of path elements. • op - the operation performed on the path (delete, replace, or update) • message - a status message
extension	Repeated field to carry gNMI extensions

7.5 gNMI Subscribe RPC

The Subscribe RPC allows you to receive updates relating to the state of data instances. The user creates a subscription using the Subscribe RPC with the desired subscription mode. The defined mode triggers how and when the data is sent to the client.

A SubscribeRequest message is sent to the target (SR Linux gNMI process gnmi_mgr) to request updates for one or more paths. A SubscribeReponse message is sent to the client over an established RPC.

7.5.1 SubscribeRequest message

The SubscribeRequest message uses the fields shown in the following table.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Table 18: SubscribeRequest fields

Field	Definition
subscribe	SubscriptionList message outlining the new set of paths to subscribe to
poll	Request for a polled update for the specified paths
extension	Repeated field to carry gNMI extensions

Subscriptions are set once and cannot be modified. A new Subscribe RPC call must be created for new paths. To end an existing subscription, the client must cancel the associated Subscribe RPC.

7.5.1.1 SubscriptionList message

A SubscriptionList message indicates a set of paths where common subscription behavior is required. The SubscriptionList message uses the fields shown in the following table.

Table 19: SubscriptionList fields

Field	Definition
subscription	A set of subscription messages indicating the paths associated with the subscription
mode	Type of subscription to create: • ONCE • STREAM (default) – ON_CHANGE – SAMPLE • sample_interval – TARGET_DEFINED • POLL
extension	Repeated field to carry gNMI extensions

7.5.1.1.1 ONCE subscription

ONCE subscriptions are one-time requests. A ONCE subscription is created by sending a SubscribeRequest message with the subscribe field containing a SubscriptionList, with the mode type set to ONCE. The relevant update messages are sent and the RPC channel is closed.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

7.5.1.1.2 STREAM subscription

STREAM subscriptions are long-lived and transmit updates indefinitely. To create a STREAM subscription, a SubscribeRequest message containing a SubscriptionList is sent and the mode type is set to STREAM. The STREAM mode subscription message also specifies one of the following modes:

- ON_CHANGE - Updates are sent when data item values change.

  The ON_CHANGE subscription can also specify a heartbeat interval, which causes the data to be sent at the specified interval whether or not the values have changed.
- SAMPLE - Data is sent at specified intervals as specified in the sample_interval field. The maximum sample rate is a 64-bit integer in nanoseconds and minimum is 0.
- TARGET_DEFINED - The target dictates the ideal subscription type to create for each leaf. For example, if the path refers to event-driven leaves, then an ON_CHANGE subscription may be created. If the data contains counters values, a SAMPLE subscription may be created.

7.5.1.1.3 POLL subscription

POLL subscriptions allow clients to regularly retrieve data items on-demand from long-lived RPCs.

To initiate a POLL subscription, the client sends an encapsulated SubscriptionList message that indicates the requested set of paths first, then a SubscribeRequest message with a poll field consisting of an empty poll message. In response, the target node generates updates for all paths defined in the SubscriptionList.

7.5.2 SubscribeResponse message

The SubscribeResponse message uses the fields shown in the following table.

Table 20: SubscribeResponse fields

Field	Definition
update OR sync_response	A response field. Only one type can be specified per message: • update - provides an update value for a subscribed data entity • sync_response - Boolean field that indicates all path data values have been transmitted at least once
extension	Repeated field to carry gNMI extensions

7.6 gNMI Capabilities RPC

The Capabilities RPC allows you to discover the capabilities of a specific gNMI server.

A CapabilityRequest message is sent by the client to request capability information from the target. The target replies with a CapabilityResponse message that includes its gNMI service version, the versioned data models it supports, and the supported data encodings.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

This information is used in subsequent RPC messages from the client to indicate the set of models that the client uses, and the encoding used for data.

7.6.1 CapabilityRequest message

The CapabilityRequest message is sent by the client to request capability information from the target. The CapabilityRequest message carries a single repeated extension field which can be used to carry gNMI extensions.

Example: CapabilityRequest message

```
message CapabilityRequest {
 repeated gnmi_ext.Extension extension = 1;
}
```

7.6.2 CapabilityResponse message

A CapabilityResponse message is sent from the target and includes the following fields:

- supported_models - a set of ModelData messages describing each model supported by the target
- supported_encodings - an enumerated field describing the data encodings supported by the target (ASCII and JSON_IETF are supported)
- gNMI_version - the version of the gNMI service supported by the target
- encoding - a repeated field for gNMI extensions

Example: CapabilityResponse message

```
message CapabilityResponse {
 repeated ModelData supported_models = 1;
 repeated Encoding supported_encodings = 2;
 string gNMIversion = 3;
 repeated gnmi_ext.Extension extension = 4;
}
```

7.7 Candidate mode

gNMI uses its own private exclusive candidate that restricts other users or services from making simultaneous changes to a configuration. If another exclusive session is already active, any attempted gNMI updates fail with an error.

The gNMI server uses the private exclusive candidate name **gnmirpc-**<*n*>, where <*n*> is a 32-bit number starting at 1 that increments for every request received, but resets on a gNMI server restart.

7.8 gNMI extensions

Each top-level RPC message (for example, the SubscribeRequest and SubscribeResponse for the Subscribe RPC) defines an extensions field that can carry additional parameters for a gNMI RPC.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

The gNMI specification provides well-known extensions within the specification itself and also allows vendors to create their own custom extensions. SR Linux supports the following well-known extensions:

- Depth
- Commit Confirmed

gNMI extensions are defined in the gnmi_ext.proto file, available here:

https://github.com/openconfig/gnmi/blob/master/proto/gnmi_ext/gnmi_ext.proto

In addition, each well-known extension has associated reference documentation, available here:

https://github.com/openconfig/reference/tree/master/rpc/gnmi

7.8.1 Commit Confirmed extension

In certain deployments, the client and server are separated by a complex network. In these cases, it is not safe to assume that the pushed configuration will not break connectivity to the network device, or that the network device has out-of-band access.

To address these issues, the Commit Confirmed extension provides a means to automatically roll back the applied configuration after a certain period has elapsed if an incorrect configuration is pushed.

Commit Confirmed is a well-known gNMI extension defined by OpenConfig in the gnmi_ext.proto, available here:

https://github.com/openconfig/gnmi/blob/v0.11.0/proto/gnmi_ext/gnmi_ext.proto#L35

The Commit Confirmed reference documentation is available here:

https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-commit-confirmed.md

SR Linux supports v0.1.0 of this extension.

7.8.2 Depth extension

To align with the principle of keeping the gNMI server implementation simple, the gNMI specification supports no subtree filtering by default. However, while simplicity is desirable, the implicit recursiveness of the requested gNMI data may be considered a limitation for some gNMI implementations.

As an alternative, the Depth gNMI extension allows a client to control the depth of the recursion when the server evaluates a group of paths in the Subscribe or Get RPC.

Orchestration, network management, and monitoring systems can benefit from this extension because it provides the following capabilities:

- reduces the load on the server when data is to be fetched from the Network OS during the recursive data extraction
- reduces the bytes on the wire payload, by sending less data

Depth is a well-known gNMI extension defined by OpenConfig in the gnmi_ext.proto, available here:

https://github.com/openconfig/gnmi/blob/v0.11.0/proto/gnmi_ext/gnmi_ext.proto#L36

The Depth reference documentation is available here:

https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-depth.md

SR Linux supports v0.1.0 of this extension.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

7.9 gNMI examples

Open source clients can be used to run GetRequests, SetRequests, subscriptions, and capabilities. The examples that follow show requests and responses using the following clients although any client that conforms to gNMI specifications can be used:

- gnmi_get — used for simple GetRequests
- gnmi_set — used for simple SetRequests
- gnmi_cli — used for SubscribeRequests, and advanced GetRequests and SetRequests
- gnmi_capabilities — used for CapabilityRequests

7.9.1 gnmi_get examples

The get gNMI-RPC allows you to retrieve state and configuration from a datastore. The following examples are shown:

- get all request
- get interface with wildcard key request

Example: get all request

```
# gnmi_get -target_addr 172.18.0.6:50052 -insecure -xpath '/'
== getRequest:
path: <
>
encoding: JSON_IETF
```

Response (get all)

```
notification: <
  timestamp: 1565672122888042050
  update: <
    path: <
    >
    val: <
      json_ietf_val: "{\n \"srl_nokia-acl:acl\": {\n \"ipv4-
      filter\" ---- snip ---- ]\n }\n}\n"
    >
  >
>
```

Example: get interface with wildcard key request

```
# gnmi_get -target_addr 172.18.0.6:50052 -insecure -xpath
'/interface[name=mgmt0]/subinterface[index=*]'
== getRequest:
path: <
  elem: <
    name: "interface"
    key: <
      key: "name"
      value: "mgmt0"
    >
  >
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
  elem: <
    name: "subinterface"
    key: <
      key: "index"
      value: "*"
    >
  >
>
encoding: JSON_IETF
```

Response (get interface with wildcard key)

```
notification: <
  timestamp: 1565671919030747121
  update: <
    path: <
      elem: <
        name: "srl_nokia-interfaces:interface"
        key: <
          key: "name"
          value: "mgmt0"
        >
      >
    >
    val: <
      json_ietf_val: "{\n \"name\": \"mgmt0\",\n \"subinterface\":
      [\n {\n \"index\": 0,\n \"admin-state\": \"enable\",\n
      \"ip-mtu\": 1500,\n \"ifindex\": 524288000,\n \"operstate\":
      \"up\",\n \"last-change\": \"2019-08-
      11T17:21:48.366Z\",\n \"ipv4\": {\n \"allow-directedbroadcast\":
      false,\n \"dhcp-client\": true,\n
      \"address\": [\n {\n \"ip-prefix\":in
      \"172.18.0.6/24\",\n \"origin\": \"dhcp\"\n }\n
      ],\n \"srl_nokia-interfaces-nbr:arp\": {\n
      \"timeout\": 14400,\n \"neighbor\": [\n {\n
      \"ipv4-address\": \"172.18.0.1\",\n \"link-layeraddress\":
      \"02:42:45:9D:DB:FC\",\n \"origin\":
      \"dynamic\",\n \"expiration-time\": \"2019-08-
      13T07:14:34.707Z\"\n },\n {\n
      \"ipv4-address\": \"172.18.0.2\",\n \"link-layeraddress\":
      \"02:42:AC:12:00:02\",\n \"origin\":
      \"dynamic\",\n \"expiration-time\": \"2019-08-
      13T05:17:51.893Z\"\n }\n ]\n }\n },\n
      \"ipv6\": {\n \"dhcp-client\": true,\n \"address\": [\n
      {\n \"ip-prefix\": \"2001:172:18::6/80\",\n
      \"origin\": \"dhcp\",\n \"status\": \"preferred\"\n
      },\n {\n \"ip-prefix\":
      \"fe80::42:acff:fe12:6/64\",\n \"origin\": \"linklayer\",\
      n \"status\": \"preferred\"\n }\n
      ],\n \"srl_nokia-interfaces-nbr:neighbor-discovery\": {\n
      \"dup-addr-detect\": true,\n \"reachable-time\": 30,\n
      \"stale-time\": 14400\n }\n },\n \"statistics\": {\n
      \"in-pkts\": \"5136\",\n \"in-octets\": \"438953\",\n
      \"in-error-pkts\": \"0\",\n \"in-discarded-pkts\": \"0\",\n
      \"in-terminated-pkts\": \"5136\",\n \"in-terminated-octets\":
      \"438953\",\n \"in-forwarded-pkts\": \"0\",\n \"inforwarded-
      octets\": \"0\",\n \"out-forwarded-pkts\":
      \"6062\",\n \"out-forwarded-octets\": \"2746613\",\n
      \"out-error-pkts\": \"0\",\n \"out-discarded-pkts\": \"0\",\n
      \"out-pkts\": \"6062\",\n \"out-octets\": \"2746520\"\n
      },\n \"srl_nokia-qos:qos\": {\n \"input\": {\n
      \"classifiers\": {\n \"ipv4-dscp\": \"default\",\n
      \"ipv6-dscp\": \"default\",\n \"mpls-tc\": \"default\"\n
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
      }\n }\n }\n }\n ]\n}\n"
    >
  >
>
```

7.9.2 gnmi_set examples

The set gNMI-RPC allows you to modify the state. The following examples are shown:

- set delete request
- set update all request

Example: set delete request

```
# gnmi_set -target_addr 172.18.0.3:50052 -username admin -password
admin -insecure -delete /system/name/host-name
== setRequest:
delete: <
  elem: <
    name: "system"
  >
  elem: <
    name: "name"
  >
  elem: <
    name: "host-name"
  >
>
```

Response (set delete)

```
response: <
  path: <
    elem: <
      name: "system"
    >
    elem: <
      name: "name"
    >
    elem: <
      name: "host-name"
    >
  >
  op: DELETE
>
timestamp: 1567203341816078044
```

Example: set an update all request

```
# gnmi_set -target_addr 172.18.0.3:50052 -username admin -password
admin -insecure -update /system/name/host-name:replaced-host -replace
/system/name/domain-name:replaced-domain
== setRequest:
replace: <
  path: <
    elem: <
      name: "system"
    >
    elem: <
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
      name: "name"
    >
    elem: <
      name: "domain-name"
    >
  >
  val: <
    string_val: "replaced-domain"
  >
>
update: <
  path: <
    elem: <
      name: "system"
    >
    elem: <
      name: "name"
    >
    elem: <
      name: "host-name"
    >
  >
  val: <
    string_val: "replaced-host"
  >
>
```

Response (set update all)

```
response: <
  path: <
    elem: <
      name: "system"
    >
    elem: <
      name: "name"
    >
    elem: <
      name: "domain-name"
    >
  >
  op: REPLACE
response: <
  path: <
    elem: <
      name: "system"
    >
    elem: <
      name: "name"
    >
    elem: <
      name: "host-name"
    >
  >
  op: UPDATE
>
timestamp: 1567204165851469784
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

7.9.3 gnmi_cli examples

The cli gNMI-RPC allows you to subscribe and receive updates on the state of a data instance. The following examples are shown:

- Subscribe - ONCE for all (one-time subscription) request
- Subscribe - STREAM ON_CHANGE interface (long term subscription) request

In these examples, `-qt` specifies the subscription type. ONCE mode is the default and therefore is not shown in the first example.

Example: Subscribe ONCE for all request

```
# gnmi_cli -a 172.18.0.6:50052 -insecure -q '/'
```

Response (subscribe ONCE for all)

```
{
  "acl": {
    "acl-filter": {
      "allow_sip_dip": {
        "type": {
          "ipv4": {
            "entry": {
              "10": {
                "action": {
                  "accept": {
                    "log": "false"
                  }
-- Snip –
}
```

Example: Subscribe STREAM ON_CHANGE interface request

```
# gnmi_cli -a 172.18.0.6:50052 -insecure --qt streaming -q
'/interface[name=mgmt0]'
```

Response (Subscribe STREAM ON_CHANGE interface)

```
{
  "interface": {
    "mgmt0": {
      "admin-state": "enable",
      "ethernet": {
      "flow-control": {
        "receive": "false"
      },
      "hw-mac-address": "02:42:AC:12:00:06",
      "statistics": {
        "in-crc-errors": "0",
        "in-fragment-frames": "0",
        "in-jabber-frames": "0",
        "in-mac-pause-frames": "0",
        "in-oversize-frames": "0",
        "out-mac-pause-frames": "0"
      }
    },
    "ifindex": "524304383",
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
    "last-change": "2019-08-30T18:44:45.490Z",
    "mtu": "1514",
    "oper-state": "up",
    "statistics": {
      "carrier-transitions": "1",
      "in-broadcast-pkts": "5",
      "in-errors": "0",
      "in-fcs-errors": "0",
      "in-multicast-pkts": "1356",
      "in-octets": "612022",
      "in-unicast-pkts": "4662",
      "out-broadcast-pkts": "1",
      "out-errors": "0",
      "out-multicast-pkts": "456",
      "out-octets": "2724476",
      "out-unicast-pkts": "5505"
    },
    "subinterface": {
      "0": {
        "admin-state": "enable",
        "ifindex": "524288000",
        "ip-mtu": "1500",
        "ipv4": {
          "address": {
            "172.18.0.6/24": {
              "origin": "dhcp"
            }
          },
          "allow-directed-broadcast": "false",
          "arp": {
            "neighbor": {
              "172.18.0.1": {
                "expiration-time": "2019-08-31T01:13:22.987Z",
                "link-layer-address": "02:42:45:9D:DB:FC",
                "origin": "dynamic"
              },
              "172.18.0.2": {
                "expiration-time": "2019-08-30T22:44:54.422Z",
                "link-layer-address": "02:42:AC:12:00:02",
                "origin": "dynamic"
              }
            },
            "timeout": "14400"
          },
          "dhcp-client": "true"
        },
        "ipv6": {
          "address": {
            "2001:172:18::6/80": {
              "origin": "dhcp",
              "status": "preferred"
            },
           "fe80::42:acff:fe12:6/64": {
              "origin": "link-layer",
              "status": "preferred"
            }
          },
          "dhcp-client": "true",
          "neighbor-discovery": {
            "dup-addr-detect": "true",
            "reachable-time": "30",
            "stale-time": "14400"
          }
        },
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
        "last-change": "2019-08-30T18:44:45.490Z",
        "oper-state": "up",
        "qos": {
          "input": {
            "classifiers": {
              "ipv4-dscp": "default",
              "ipv6-dscp": "default",
              "mpls-tc": "default"
            }
          }
        },
        "statistics": {
          "in-discarded-pkts": "0",
          "in-error-pkts": "0",
          "in-forwarded-octets": "0",
          "in-forwarded-pkts": "0",
          "in-octets": "404380",
          "in-pkts": "4679",
          "in-terminated-octets": "404380",
          "in-terminated-pkts": "4679",
          "out-discarded-pkts": "0",
          "out-error-pkts": "0",
          "out-forwarded-octets": "2409995",
          "out-forwarded-pkts": "5511",
          "out-octets": "2409995",
          "out-pkts": "5511"
        }
      }
    },
    "vlan-tagging": "false"
  }
}
{
  "interface": {
    "mgmt0": {
      "statistics": {
        "in-octets": "615366"
      }
    }
  }
}
{
  "interface": {
    "mgmt0": {
      "statistics": {
        "in-unicast-pkts": "4693"
      }
    }
  }
}
{
  "interface": {
    "mgmt0": {
      "statistics": {
        "out-octets": "2736287"
      }
    }
  }
}
.
.
.
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

7.9.4 gnmi_capabilities example

The capabilities gNMI-RPC allows you to discover the capabilities of a specific gNMI server. The following example shows a request to obtain model, data encodings, and version for a specified server.

Example: Request server capabilities for specified server

```
gnmi_capabilities   -username admin -password admin --target_addr [172.18.0.8]:50264
```

Response (request server capabilities)

```
supported_models: <
 name: "urn:srl_nokia/aaa:srl_nokia-aaa"
 organization: "Nokia"
 version: "2020-12-31"
>

supported_models: <
 name: "urn:srl_nokia/aaa-types:srl_nokia-aaa-types"
 organization: "Nokia"
 version: "2019-11-30"
>
.
.
.
supported_encodings: JSON_IETF
supported_encodings: ASCII
supported_encodings: PROTO
supported_encodings: 45
supported_encodings: 44
supported_encodings: 46
supported_encodings: 47
gNMI_version: "0.7.0"
```

7.10 gNMI service configuration

SR Linux supports a gRPC server that allows external gRPC clients, including gNMI clients, to connect to the device and modify the configuration and collect state information.

See the "Management servers" chapter in the *SR Linux Configuration Basics Guide* for information about how to configure the gRPC server for gNMI support.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

8 gNOI

gRPC Network Operations Interface (gNOI) defines a set of gRPC-based services for executing operational commands on network devices. The individual RPCs and messages that perform the gNOI operations required on the node are defined at the following location: https://github.com/openconfig/gnoi. This repository also stores the various per-service protos in subdirectories.

The .proto gNOI files for SR Linux are stored in `/opt/srlinux/protos/gnoi`. These protos allow you to verify supported proto versions for each release, as gRPC services do not support direct version querying.

SR Linux supports the following gNOI services:

- gNOI OS service
- gNOI BootConfig service
- gNOI FactoryReset service
- gNOI File service
- gNOI System service
- gNOI Healthz service
- gNOI Packet Link Qualification service
- gNOI BGP service
- gNOI Containerz service

8.1 gNOI OS service

The gNOI OS service provides an interface to install an OS package on a target node. SR Linux supports the gNOI OS service on both the active and standby CPMs (referred to as supervisors in gNOI).

To perform the OS installation, the client progresses through the following three gNOI OS RPCs:

- Install RPC
- Activate RPC
- Verify RPC

The protos used to define the OS service were pulled from the following hash: https://github.com/openconfig/gnoi/commit/93cdd9ae9f35d8b4bc1599d0a727b294faeca352.

8.1.1 Install RPC

The Install RPC transfers the OS package to the target node. The target node first attempts to copy the specified OS package between the CPMs before it accepts the transfer from the client.

To refer to the OS version, SR Linux uses the same string in gNOI as the one used with ZTP (*version string-build number*) , for example: `v22.11.1-010`. The download folder for the OS is located at: `/var/`

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

`run/srlinux/gnoi`. To validate that the transferred OS package is valid and bootable before installation, the platform performs a hash check against the md5sum that is embedded in the .bin file.

On a dual CPM node, only the active CPM runs the gNOI service. The Install RPC transfers the OS to the active CPM.

Note: SR Linux does not support the **standby_supervisor** option. On a dual CPM node, the transferred image is synced automatically to the standby CPM using ZTP.

One Install RPC is required for each CPM. Concurrent Install RPCs are not allowed on the same target node.

Install RPC structure

```
rpc Install(stream InstallRequest) returns (stream InstallResponse);
```

8.1.2 Activate RPC

The Activate RPC sets the requested OS version for the target node to use at the next reboot. It also reboots the target node if the `no_reboot` flag is not set.

Note: If the requested image fails to boot, SR Linux cannot attempt to boot a secondary image. In this case, the system can revert to the rescue image.

On a dual CPM node, if you perform this RPC on the active CPM, it triggers a switchover to the standby CPM before rebooting the previously active CPM.

Activate RPC structure

```
rpc Activate(ActivateRequest) returns (ActivateResponse);
```

8.1.3 Verify RPC

The Verify RPC checks the OS version running on the target node. The client can call this RPC multiple times while the target node boots until the activation is successful.

Note: The activation_fail_message is not supported because if the target node does not boot, it remains in a failure state and does not revert to a previous version of OS.

Verify RPC structure

```
rpc Verify(VerifyRequest) returns (VerifyResponse);
```

8.2 gNOI BootConfig service

The gNOI BootConfig service provides an interface to update the boot configuration on a target node similar to the Bootz protocol. This service is typically used after completion of the Bootz workflow.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

For more information about Bootz with SR Linux, see the *SR Linux Software Installation Guide*.

SR Linux supports the following gNOI BootConfig RPCs:

- GetBootConfig RPC
- SetBootConfig RPC

8.2.1 GetBootConfig RPC

The GetBootConfig RPC allows the client to retrieve the current boot configuration from the target node.

GetBootConfig RPC structure

```
rpc GetBootConfig(GetBootConfigRequest) returns (GetBootConfigResponse) {}
```

8.2.2 SetBootConfig RPC

The SetBootConfig RPC allows the client to replace the current boot configuration on the target node with the new configuration provided.

SetBootConfig RPC structure

```
rpc SetBootConfig(SetBootConfigRequest) returns (SetBootConfigResponse) {}
```

8.3 gNOI FactoryReset service

The FactoryReset service enables gNOI clients to reset a target node to boot using a golden image and to optionally format persistent storage.

One of the practical applications of this service is the ability to factory reset a device before performing Zero Touch Provisioning (ZTP).

SR Linux supports the following gNOI FactoryReset RPC:

- Start RPC

The protos used to define the FactoryReset service were pulled from the following hash: v0.1.0.

8.3.1 Start RPC

The Start RPC allows the client to instruct the target node to immediately clean all existing state data (including storage, configuration, logs, certificates, and licenses) and boot using the OS image configured as the golden image. The golden image is the image that the device resets to when a factory reset is performed. You can set the golden image using the **tools system boot golden-image** command. To view the available images to select from, use the **tools system boot available-images** command. If the golden image is not set, the system boots using the current OS image.

The Start RPC supports optional flags to:

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- roll back to the OS configured as the golden image
- zero-fill any state data saved in persistent storage

If the golden image is configured and the **factory_reset** flag is set to **true**, SR Linux resets to the golden image. If the **factory_reset** flag is omitted or set to **false**, SR Linux boots using the current running image, but all existing state data is cleaned.

If any optional flags are set but not supported, the target node returns a gRPC Status message with code INVALID_ARGUMENT with the details value set to the appropriate ResetError message.

Start RPC structure

```
rpc Start(StartRequest) returns (StartResponse);
```

8.3.2 gNOI FactoryReset tools CLI command

To allow the gNOI FactoryReset behavior to be invoked using the CLI, SR Linux supports the following tools command:

tools system factory-reset reboot [factory-os] [zero-fill]

where:

- **factory-os** — rolls the system back to the OS image configured as the golden image
- **zero-fill** — performs a zero-fill of the permanent storage where state data is stored

8.4 gNOI File service

The gNOI File service allows the client to transfer files to and from the target node. The main use for this service is extracting debugging information through the transfer of system logs and core files.

SR Linux supports the following gNOI File RPCs:

- Get RPC
- Put RPC
- Stat RPC
- Remove RPC

Note: The TransferToRemote RPC is not supported.

The protos used to define the gNOI File service were pulled from the following hash: https://github.com/openconfig/gnoi/commit/93cdd9ae9f35d8b4bc1599d0a727b294faeca352.

8.4.1 Get RPC

The Get RPC reads and streams the contents of a file from a target node to the client using sequential messages, and sends a final message containing the hash of the streamed data before closing the stream.

The target node returns an error if:

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- An error occurs while reading the file.
- The file does not exist.

Get RPC structure

```
rpc Get(GetRequest) returns (stream GetResponse) {}
```

8.4.2 Put RPC

The Put RPC streams data to the target node and writes the data to a file. The client streams the file using sequential messages. The initial message contains information about the filename and permissions. The final message includes the hash of the streamed data.

The target node returns an error if:

- An error occurs while writing the data.
- The location does not exist.

Put RPC structure

```
rpc Put(stream PutRequest) returns (PutResponse) {}
```

8.4.3 Stat RPC

The Stat RPC returns metadata about files on the target node.

If the path specified in the StatRequest references a directory, the StatResponse returns the metadata for all files and folders, including the parent directory. If the path references a direct path to a file, the StatResponse returns metadata for the specified file only.

The target node returns an error if:

- The file does not exist.
- An error occurs while accessing the metadata.

Stat RPC structure

```
rpc Stat(StatRequest) returns (StatResponse) {}
```

8.4.4 Remove RPC

The Remove RPC removes the specified file from the target node.

The target node returns an error if:

- An error occurs during the remove operation (for example, permission denied).
- The file does not exist.
- The path references a directory instead of a file.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Remove RPC structure

```
rpc Remove(RemoveRequest) returns (RemoveResponse) {}
```

8.5 gNOI System service

The gNOI System service defines an interface that allows a client to perform operational tasks on target network nodes. SR Linux supports the following gNOI System RPCs:

- Ping RPC
- Traceroute RPC
- Time RPC
- SwitchControlProcess RPC
- Reboot RPC
- CancelReboot RPC
- RebootStatus RPC
- KillProcess RPC

The protos used to define the gNOI System service were pulled from the following hash: https://github.com/openconfig/gnoi/commit/93cdd9ae9f35d8b4bc1599d0a727b294faeca352.

8.5.1 Ping RPC

The Ping RPC allows the client to execute the ping command on the target node. The target node streams the results back to the client. Some targets do not stream any results until they receive all results. If the RPC does not specify a packet count, the ping operation uses a default of five packets.

Note:

- The Ping RPC does not currently support specification of a network-instance. The ping is executed in the network-instance where the gNMI server is running.
- SR Linux does not support setting the interval field in the PingRequest to -1 (flood ping).

Ping RPC structure

```
rpc Ping(PingRequest) returns (stream PingResponse) {}
```

8.5.2 Traceroute RPC

The Traceroute RPC allows the client to execute the traceroute command on the target node. The target node streams the results back to the client. Some targets do not stream any results until they receive all results. If the RPC does not specify a hop count, the traceroute operation uses a default of 30.

Note:

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- The Traceroute RPC does not currently support specification of a network-instance. The traceroute is executed in the network-instance where the gRPC server is running.
- In the TracerouteRequest, SR Linux does not support the TCP and UDP enum values for the l4protocol field. Only ICMP is supported.
- In the TracerouteResponse, SR Linux does not support the mpls and as_path fields.

Traceroute RPC structure

```
rpc Traceroute(TracerouteRequest) returns (stream TracerouteResponse) {}
```

8.5.3 Time RPC

The Time RPC returns the current time on the target node. It is typically used to test whether the target is currently responding.

Time RPC structure

```
rpc Time(TimeRequest) returns (TimeResponse) {}
```

8.5.4 SwitchControlProcessor RPC

The SwitchControlProcessor RPC switches the active control processing module (CPM) on the target node to the control slot (A or B) that is specified in the request message.

SwitchControlProcessor RPC structure

```
rpc SwitchControlProcessor(SwitchControlProcessorRequest)
    returns (SwitchControlProcessorResponse) {}
```

8.5.5 Reboot RPC

The Reboot RPC allows the client to reboot a target node, either immediately or at some time in the future. It triggers the reboot of the entire chassis. It also supports specification of a reboot method (for example, cold or warm reboot), however, if the target node does not support the specified reboot method, the Reboot RPC fails.

Note: SR Linux supports only the cold reboot method, and does not support rebooting of subcomponents.

If a reboot is pending on the active control processor, the service rejects all other reboot requests.

Reboot RPC structure

```
rpc Reboot(RebootRequest) returns (RebootResponse) {}
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

8.5.6 CancelReboot RPC

The CancelReboot RPC allows the client to cancel any pending reboot requests on the target node.

Note: SR Linux does not support canceling a reboot for a subcomponent.

CancelReboot RPC structure

```
message CancelRebootResponse { }
```

8.5.7 RebootStatus RPC

The RebootStatus RPC allows the client to query the status of a reboot on the target node.

Note: SR Linux does not support querying on a single component at a time.

RebootStatus RPC structure

```
rpc RebootStatus(RebootStatusRequest) returns (RebootStatusResponse) {}
```

8.5.8 KillProcess RPC

The KillProcess RPC allows a client to kill an OS process and optionally restart it on the target node.

To specify the process to kill, the RPC must match the application name referenced in the **tools system app-management application** <*name*> command.

Mapping of termination signals to SR Linux commands

The KillProcess RPC termination signals map to SR Linux commands as follows:

Table 21: Mapping of termination signals to SR Linux commands

Termination signal	SR Linuxcommand	Command if restart field is true
SIGNAL_TERM	**stop**	**restart** (this option runs a warm or cold restart based on what is supported)
SIGNAL_KILL	**kill**	**restart cold**
SIGNAL_HUP	**reload**	—

KillProcess RPC structure

```
rpc KillProcess(KillProcessRequest) returns (KillProcessResponse) {}
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

8.6 gNOI Healthz service

To align with general design principles of distributed systems, the gNOI Healthz service allows system components to report their own health.

Debug commands can display details about the health and state of a component. The Healthz service exposes these interfaces as queryable endpoints. In doing so, it allows clients to validate the health of components and, if unhealthy, gather device-specific data to help triage or reproduce issues.

The Healthz service allows a client to initiate health checks on a target node using the Check RPC. Alternatively, the target node can self-initiate the check and report the results to the client.

A client can then use the List or Get RPC to retrieve health events associated with the affected component and subcomponents. These health events are included in ComponentStatus messages and can be helpful to debug or further root cause the reported fault.

As part of the event response, the List or Get RPC can identify specific artifacts associated with the event, which the client can then retrieve using the Artifact RPC. The client can also call the Acknowledge RPC to acknowledge the retrieval of an event (corresponding to a series of artifacts). By default, acknowledged events are no longer included in the list of events.

The SR Linux components that support some degree of Healthz are as follows (listed in native schema):

- `.platform.control{}`
- `.platform.linecard{}`
- `.platform.chassis`
- `.platform.fan-tray{}`
- `.platform.power-supply{}`
- `.platform.fabric{}`
- `.platform.linecard{}.forwarding-complex{}`
- `.interface{}.transceiver`

This includes all control, linecard, and fabric modules, along with power supplies and fans, individual transceivers and the chassis itself. Software components, such as routing protocol daemons, are not yet supported with the gNOI Healthz service.

SR Linux supports the following gNOI Healthz RPCs:

- Check RPC
- Get RPC
- List RPC
- Acknowledge RPC
- Artifact RPC

SR Linux uses v1.3.0 of the gNOI Healthz service protos, pulled from the following hash: https://github.com/openconfig/gnoi/blob/4f5cb0885a26a52f9c30acc236d307192c665bd8/healthz/healthz.proto.

Collection of artifacts

The workflow for collecting gNOI Healthz artifacts is as follows:

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- When a system component becomes unhealthy, the system transmits health state information via telemetry that indicates the `healthz/state/status` of the component has transitioned to UNHEALTHY.
- When the client observes the transition to UNHEALTHY, it can call the Get or List RPCs to collect the events that occurred on the component.
- As the collection of some artifacts can be service impacting, all artifacts are not always automatically collected for an event. In this case, the client can call the Check RPC to collect the additional service impacting artifacts. This provides an opportunity to coordinate the collection of these artifacts when the operational risk of doing so is minimized (for example, by first removing traffic from the target node). To refer to a previously reported event, the Check RPC request must populate the **event_id** field with the ID reported in a prior Get or List response. After this Check RPC call, the client can call the Get or List RPCs to obtain the additional artifacts collected for the specified event.
- If a component returns to a healthy status, the system sends updated telemetry information to ensure that the external clients are updated about the current health status, even if the clients make no additional Healthz calls to the system.

Healthz events persistence

Healthz events created for the components are written to disk and persist across restarts of the service or software components.

SR Linux saves Healthz events in the `/etc/opt/srlinux/gnoi/healthz/events` directory, and rotates the event files to prevent overflow of the partition size. The rotation limit is set to 10 MB for all events combined.

During an unexpected CPM failover Healthz event, the creation and storage of events and artifacts can be interrupted by a CPM switchover. In this case, defer any user-initiated CPM switchover while the system is still processing the Healthz events and artifacts.

Healthz events are written to disk in intervals (every minute) to mitigate high disk pressure for frequently changing events (for example, a flapping interface).

gNMI component paths

The Healthz service works in conjunction with telemetry streamed via gNMI. The system can stream OpenConfig or native YANG paths for a specific component when the component becomes unhealthy.

To maintain Healthz parameters, SR Linux includes a **healthz** container for each of the supported components. For example, the following container maintains Healthz data for the control module:

```
augment /srl-platform:platform/srl-platform-control:control:
    +--ro healthz
        +--ro status? enumeration
        +--ro last-unhealthy? srl-comm:date-and-time-delta
        +--ro unhealthy-count? srl-comm:zero-based-counter64
```

When the Healthz service references a gNMI path (`gnoi.types.Path`), it specifies the complete path to a component, for example: `/components/component[name=FOO]`.

8.6.1 Check RPC

The Check RPC allows a client to execute a set of validations against a component. As with other Healthz operations, the component is specified using its gNMI path.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

The Check RPC produces a Healthz ComponentStatus message, which contains a list of the artifacts generated from the validation process.

While the system can initiate health checks itself, these checks are limited to operations that do not impact the device functionality. Checks that are potentially service impacting require use of the Check RPC.

Note: Nokia recommends the implementation of command authorization to restrict use of these commands to prevent running unauthorized Check RPCs during normal operations.

The CheckRequest message includes an optional **event_id** field. When populated, this field directs the system to perform the check for a prior event. In this case, the device collects artifacts that were not collected automatically when the event occurred (to prevent service impacts). The collected artifacts are returned in the artifact list for the event in subsequent Get or List RPC calls.

A CheckRequest for a previous **event_id** does not overwrite previous artifacts that were collected at the time of the event.

Check RPC structure

```
rpc Check(CheckRequest) returns (CheckResponse) {}
```

8.6.2 Get RPC

After a health check, the client can use the Get (or List) RPC to retrieve the health events that are associated with a component.

The Get RPC retrieves the latest health event for the specified component. Each event consists of a collection of data that you can use to debug or root cause the fault. Unlike the List RPC, the Get RPC returns only the latest event.

The GetResponse returns a ComponentStatus message that corresponds to the latest health event for the component and each of its subcomponents. As a result, the Get RPC can return multiple ComponentStatus messages for a single component.

Each ComponentStatus message includes a set of ArtifactHeader messages that correspond to the health event, and provide identifiers and types for the artifacts returned by the system. All artifacts listed within the same ComponentStatus message share the same acknowledgement state and expiry time.

When a client invokes a Get RPC on a path, this action is not recorded as an event for this path and no health checks are performed.

Get RPC structure

```
rpc Get(GetRequest) returns (GetResponse) {}
```

8.6.3 List RPC

As an alternative to the Get RPC, the client can use the List RPC to retrieve not just the latest but all health events for the specified component and its subcomponents. Similar to the Get RPC, the List RPC also returns a series of ComponentStatus messages, which have the same semantics as those returned by the Get RPC.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

By default, events that are already acknowledged are not returned.

List RPC structure

```
rpc List(ListRequest) returns (ListResponse) {}
```

8.6.4 Acknowledge RPC

A client can use the Acknowledge RPC to indicate to the target node that the client retrieved a particular (component, event) tuple. To ensure that Healthz artifact storage does not cause resource exhaustion, SR Linux can remove saved artifacts, starting with acknowledged artifacts first.

Acknowledge RPC structure

```
rpc Acknowledge(AcknowledgeRequest) returns (AcknowledgeResponse) {}
```

8.6.5 Artifact RPC

The Artifact RPC allows a client to retrieve specific artifacts that are related to an event that the target node reported in a prior List or Get response.

Because these artifacts can be large, the Artifact RPC is implemented as a server-side streaming RPC. The Artifact RPC ensures that a target node sends these potentially large artifacts only when the client explicitly requests them.

Artifacts can be core files, dumps of state (**info from state** on the specified component), or other log files. The collection of **info from state** artifacts results in the capture of any failure reasons from either the **oper-reason** or **oper-down-reason** fields.

The client can acknowledge a retrieved event corresponding to a series of artifacts. Acknowledged events are no longer returned in the list of events by default.

Events persist across restarts of the system or its hardware and software components, and they are removed only for resource management purposes. SR Linux can use the acknowledged status to remove artifacts that are no longer relevant and, if necessary, remove artifacts that are not yet acknowledged.

Artifact RPC structure

```
rpc Artifact(ArtifactRequest) returns (stream ArtifactResponse) {}
```

8.6.6 gNOI Healthz CLI commands

To allow the gNOI Healthz events to be cleared using the CLI or by a gNMI or JSON-RPC client, SR Linux supports the following CLI command:

```
tools system grpc-server testing gnoi healthz [<component>] clear
```

You can omit the component value to clear all statistics for all components, or you can use one of the following parameters to clear statistics for the specified component only:

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- **chassis**: chassis component
- **control slot** <*id*>: control module component
- **fabric slot** <*id*>: fabric module component
- **fan-tray id** <*id*>: fan component
- **linecard slot** <*id*>: line card component
- **power-supply id** <*id*>: power supply component
- **transceiver interface** <*name*>: transceiver component

8.7 gNOI Packet Link Qualification service

The gNOI Packet Link Qualification service allows a client to validate the quality of the link between interfaces on two devices. This service defines a protocol to support the generation of packets from one device (the generator) to a peer device (the reflector). The reflector unconditionally loops back any packets it receives from the generator, and the service validates that those packets are sent and received. After the qualification is complete, the interfaces are restored to their previous state.

SR Linux uses v1.1.0 of the gNOI Packet Link Qualification service protos, pulled from the following hash: https://github.com/openconfig/gnoi/blob/671c4286820310a6fa7b016124ea248bf5cfbe76/packet_link_qualification/packet_link_qualification.proto.

The service allows for different generation and reflection modes of operation based on the hardware capabilities of the devices, however SR Linux only supports a subset of those modes. Regardless of the modes selected, a standard report is generated that upstream services can use to aggregate network-wide link quality.

Platform support

The gNOI Packet Link Qualification service is supported on 7250 IXR platforms.

Generation modes

The service supports the following packet generation modes:

- **Packet Generators**: The preferred mode for the generation of frames. This mode provides the most flexibility regarding packet rates, packet sizes, and packet content.
- **Packet Injectors (not supported on SR Linux)**: For devices that do not support a built-in packet generator.

Reflector modes

The service supports the following packet reflector modes:

- **ASIC Loopback**: The preferred mode of reflection. This mode allows for the loopback of frames to occur in the forwarding path of the device. It enables the most comprehensive testing results as all counters may be available to the qualification.
- **PMD Loopback (not supported on SR Linux)**: For devices that do not support ASIC loopback mode.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Operational status of interfaces during qualification

During the link qualification, the operational status of the interfaces displays as `testing`. After the qualification is complete, the operational status is restored to the previous state. In SR Linux, a port in `testing` status cannot be configured. Upper layer protocols consider an interface in `testing` state as operationally down. Control plane packets stop generating on the interface, but resume when the operational status moves back to `up`. In the test mode, both generating and reflecting ports indicate their status using port LEDs normally, as when forwarding traffic.

Retention and persistence of qualification results

SR Linux stores up to ten of the latest qualification results per interface; the oldest results are automatically overwritten. The results do not persist across reboots of the device nor do they persist when the warm restart is issued. During warm or cold restarts the in-progress and scheduled tests are canceled.

Supported RPCs

SR Linux supports the following gNOI Packet Link Qualification RPCs:

- Capabilities RPC
- Create RPC
- Get RPC
- Delete RPC
- List RPC

8.7.1 Service Call Flow

The following steps show a typical call flow using the gNOI Packet Link Qualification RPCs. In these examples, the client is referred to as the orchestrator.

1. Determine peer capabilities using the Capabilities RPC

```
Orchestrator                              Generator                              Reflector
| ------------- Capabilities() -------------> |                                        |
| <------------CapbilityResponse ------------ |                                        |
| ------------------------------------ Capabilities() ----------------------------> |
| <---------------------------------- CapabilityResponse() ------------------------- |
```

2. Specify generator and reflector types and interfaces to qualify using the Create RPC

```
Orchestrator                              Generator                              Reflector
| -------------------- Create() ------------> |                                        |
| ---------------------------------------- Create() --------------------------------> |
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
   *Generator and Reflector both validate the CreateRequest*
```

3. Loop Get RPC to return the status of the qualification ID until qualification is complete

```
Orchestrator                              Generator                                Reflector
| --------------------- Get() -------------> |                                         |
| <---------------- GetResponse ------------ |                                         |
| ---------------------------------------- Get() ----------------------------------> |
| <---------------------------------- GetResponse---------------------------------- |
   *Loop until qualification complete*
```

4. Retrieve results for the qualification ID using the List RPC

```
Orchestrator                              Generator                                Reflector
| --------------------- List() -------------> |                                         |
| <---------------- ListResponse ------------ |                                         |
| ---------------------------------------- List() ---------------------------------> |
| <---------------------------------- ListResponse---------------------------------- |
```

5. Delete results as required using the Delete RPC

```
Orchestrator                              Generator                                Reflector
| ------------------ Delete() -------------> |                                         |
| ---------------------------------------- Delete() ---------------------------------> |
```

8.7.2 Capabilities RPC

The Capabilities RPC allows the client to determine the generation and reflection capabilities of target peers in preparation for a packet link qualification between the two devices.

Capabilities RPC structure

```
rpc Capabilities(CapabilitiesRequest) returns (CapabilitiesResponse);
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

8.7.3 Create RPC

The Create RPC initiates a qualification operation for each interface. If it fails to create the qualification, the RPC returns an error.

When the Create RPC is called, the device interfaces are put into a forwarding mode which must not contain other traffic, either control or data. This setting can cause the generator or reflector endpoint to become unreachable.

An interface that is configured but not present in the state datastore is not eligible for testing, and `interface not found` is returned. An interface that is present in state but not in the configuration is eligible for testing.

An attempt to create a test with a duplicate ID returns an `AlreadyExists` error. This is applicable both to a test that is still running or to a completed test that is not yet deleted using the Delete RPC.

Create RPC structure

```
rpc Create(CreateRequest) returns (CreateResponse);
```

8.7.4 Get RPC

The Get RPC allows the client to retrieve the status for the specified qualification IDs.

Get RPC structure

```
rpc Get(GetRequest) returns (GetResponse);
```

8.7.5 Delete RPC

The Delete RPC allows the client to remove the qualification results for the specified qualification IDs. If a qualification is still in progress, the qualification is first canceled before it is deleted. If a qualification cannot be stopped or deleted, the RPC returns an error.

A client can cancel a running qualification early for several reasons, such as:

- Determination that the errors have already exceeded a threshold
- Desire to change the rate or MTU for a running test
- Realization that the qualification was scheduled at the wrong time

Delete RPC structure

```
rpc Delete(DeleteRequest) returns (DeleteResponse);
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

8.7.6 List RPC

The List RPC allows the client to retrieve the current results for the specified qualification IDs from the generator and reflector nodes.

List RPC structure

```
rpc List(ListRequest) returns (ListResponse);
```

8.7.7 Packet Link Qualification CLI commands

To allow the gNOI Packet Link Qualification behavior to be invoked in the CLI or by a gNMI or JSON-RPC client, SR Linux supports the following CLI commands:

- **system packet-link-qualification profile** <*profile*>
- **tools interface** <*name*> **packet-link-qualification start id** <*id*> **qualification-profile** <*profile*>
- **info from state interface** <*name*> **packet-link-qualification result** <*id*>
- **tools interface** <*name*> **packet-link-qualification cancel id** <*id*>

8.7.7.1 Configuring a gNOI packet link qualification profile using the CLI

Procedure

As an alternative to using the gNOI RPCs, you can also configure a packet link qualification profile using the **system packet-link-qualification profile** CLI command.

Example: Configure a packet link qualification profile (generator)

```
--{ + candidate shared default }--[  ]--
# info with-context system packet-link-qualification profile p1
    system {
        packet-link-qualification {
            profile p1 {
                rpc {
                    pre-sync-duration 10
                    setup-duration 20
                    duration 200
                    post-sync-duration 5
                    teardown-duration 15
                }
                packet-generator {
                    packet-rate 152737
                    packet-size 8184
                }
            }
        }
    }
```

Example: Configure a packet link qualification profile (reflector)

```
--{ + candidate shared default }--[  ]--
# info with-context system packet-link-qualification profile p1
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
    system {
        packet-link-qualification {
            profile p1 {
                rpc {
                    pre-sync-duration 10
                    setup-duration 20
                    duration 200
                    post-sync-duration 20
                    teardown-duration 15
                }
                asic-loopback {
                }
            }
        }
    }
```

8.7.7.2 Running a packet link qualification test using the CLI

Procedure

You can run a gNOI packet link qualification test using the **tools interface packet-link-qualification start id qualification-profile** CLI command and obtain results of the test using the **info from state** command. To stop or cancel the test, use the **tools interface packet-link-qualification cancel id** command.

Example: Start a packet link qualification test on the generator

```
--{ + candidate shared default }--[  ]--
# tools interface ethernet-5/4 packet-link-qualification start id id1 qualification-
profile p1
```

Example: Start a packet link qualification test on the reflector

```
--{ + candidate shared default }--[  ]--
# tools interface ethernet-8/4 packet-link-qualification start id id1 qualification-
profile p1
```

Example: Obtain test results from the generator

```
--{ + candidate shared default }--[  ]--
# info from state with-context interface ethernet-5/4 packet-link-qualification result 1
    interface ethernet-5/4 {
        packet-link-qualification {
            result 1 {
                oper-state completed
                packets-sent 30545942
                packets-received 30545942
                packets-error 0
                packets-dropped 0
                start-time "2024-02-20T16:05:03.050Z (an hour ago)"
                end-time "2024-02-20T16:08:23.050Z (an hour ago)"
                expected-rate 1249999608
                qualification-rate 1249940692
            }
        }
    }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: Obtain test results from the reflector

```
--{ + candidate shared default }--[  ]--
# info with-context from state interface ethernet-8/4 packet-link-qualification result 1
    interface ethernet-8/4 {
        packet-link-qualification {
            result 1 {
                oper-state completed
                packets-sent 30599067
                packets-received 30599067
                packets-error 0
                packets-dropped 0
                start-time "2024-02-20T16:05:02.969Z (an hour ago)"
                end-time "2024-02-20T16:08:22.969Z (an hour ago)"
            }
        }
    }
```

Example: Stop the test on the generator

```
--{ + candidate shared default }--[  ]--
# tools interface ethernet-5/4 packet-link-qualification cancel id id1
```

Example: Stop the test on the reflector

```
--{ + candidate shared default }--[  ]--
# tools interface ethernet-8/4 packet-link-qualification cancel id id1
```

8.8 gNOI BGP service

The gNOI BGP service provides a single RPC that allows the client to clear a specific BGP neighbor in one of three ways:

- hard reset
- soft reset with a route-refresh message
- soft reset (currently not supported in SR Linux)

The following shows the service definition of the gNOI BGP service.

```
service BGP {
  rpc ClearBGPNeighbor(ClearBGPNeighborRequest)
    returns (ClearBGPNeighborResponse) {}
}

message ClearBGPNeighborRequest {
  string address = 1;
  string routing_instance = 2;
  enum Mode {
    SOFT = 0;
    SOFTIN = 1;
    HARD = 2;
  }
  Mode mode = 3;
}

message ClearBGPNeighborResponse {
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
}
```

The following table displays the SR Linux command equivalents to the **ClearBGPNeighbor** RPC modes.

Table 22: ClearBGPNeighborRequest mapping to SR Linux tools command

Mode	SR Linux tools command
SOFT	`tools network-instance protocols bgp neighbor soft-clear`
HARD	`tools network-instance protocols bgp neighbor reset-peer`

8.9 gNOI Containerz service

The gNOI ContainerZ service provides a gRPC interface to manage containers, supporting the deployment of custom third-party containerized applications on network devices. For the Containerz protobuf, see:

https://github.com/openconfig/gnoi/blob/main/containerz/containerz.proto

Containerz provides the ability to deploy third-party containerized applications onto an SR Linux device. Use this functionality with caution. If used incorrectly, it may render the system inoperable.

Containerz creates a private container registry on the network device to store container images for use by the devices container management system. Containers can be started on specific CPM modules using these container images.

SR Linux supports the following gNOI Containerz RPCs:

- Deploy RPC
- ListImage RPC
- RemoveImage RPC
- RemoveContainer RPC
- ListContainer RPC
- StartContainer RPC
- StopContainer RPC
- UpdateContainer RPC
- Log RPC
- CreateVolume RPC
- RemoveVolume RPC
- ListVolume RPC
- StartPlugin RPC
- StopPlugin RPC
- ListPlugins RPC
- RemovePlugin RPC

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

8.9.1 Deploy RPC

The Deploy RPC loads the specified container image to the container registry of the target node. If the specified container image is a plugin, it is not loaded into the registry until the StartPlugin RPC is called.

SR Linux suppports uploads of image up to two GB in size. No limit exists on the total disk space that the container registry can use. You must closely monitor the available space per partition to ensure sufficient space remains. If the required disk space is unavailable, the deployment fails.

SR Linux deploys independent container registries per CPM. The Containerz interface replicates all RPCs targeting the system to both CPMs to ensure the device is ready to support CPM switchovers. For example, issuing the Deploy RPC for an image on a dual CPM system places the image in the container registry on CPM A and the container registry on CPM B. The underlying filesystem backing the container registries is not replicated and remains tied to a specific CPM.

Deploy RPC structure

```
rpc Deploy(stream DeployRequest) returns (stream DeployResponse) {}
```

8.9.2 ListImage RPC

The ListImage RPC returns all images from the active CPM on the target node that match the filter specified in the ListImageRequest message.

SR Linux responds with a list of the images, their ID, and their tags.

ListImage RPC structure

```
rpc ListImage(ListImageRequest) returns (stream ListImageResponse) {}
```

8.9.3 RemoveImage RPC

The RemoveImage RPC removes the image that matches the specified name and tag from the target node. To perform a force delete, set **force** to **True** in the RemoveImageRequest message.

RemoveImage RPC structure

```
rpc RemoveImage(RemoveImageRequest) returns (RemoveImageResponse) {}
```

8.9.4 RemoveContainer RPC

The RemoveContainer RPC removes the specified container from the target node. If the specified container is running, it is not removed unless the **force** option is set to **True** in the RemoveContainerRequest message.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

RemoveContainer RPC structure

```
rpc RemoveContainer(RemoveContainerRequest) returns (RemoveContainerResponse) {}
```

8.9.5 ListContainer RPC

The ListContainer RPC returns all containers from the target node that match the filter specified in the ListContainerRequest message. SR Linux returns only running containers unless the **all** option is set to **true** in the ListContainerRequest message.

When a container is started on SR Linux using the **location** option, the ListContainerResponse message includes the following labels:

- **net.openconfig.containerz.location.intended**: The **location** field specified at deployment
- **net.openconfig.containerz.location.actual**: The current location of the container on the operational system
- **net.openconfig.containerz.location.reason**: If the container failed to be placed, the reason for the failure

ListContainer RPC structure

```
rpc ListContainer(ListContainerRequest) returns (stream ListContainerResponse) {}
```

8.9.6 StartContainer RPC

The StartContainer RPC starts the specified container on the target node. This RPC instantiates a container on an SR Linux device. The container is created using an image already available in the SR Linux container registry (using the Deploy RPC).

To start a container, provide the **image_name**, image **tag** and an optional **instance_name** in the StartContainerRequest message.

If no **instance_name** is provided, SR Linux assigns one. When an instance (identified by an **instance_name**) already exists, the StartContainer RPC returns an error if the container is running, or if it is stopped but the StartContainerRequest contains more fields than just the **instance_name**. If the existing container is stopped and the StartContainerRequest contains only the **instance_name**, the system starts the container.

Common additional parameters for the StartContainerRequest message include:

- **ports**: Multiple ports may be specified. Each port has an **internal** (container-side) port and an **external** (network-facing) port.
- **environment**: Each environment variable is a map of two strings, the first being the key and the second being the value.
- **volumes**: Storage volumes created using the CreateVolume RPC (see the CreateVolume RPC section for further details)
- **network**: The network the container connects to. Supported options include **host** and **bridged**. The **network** field also accepts any externally created network name, but the creation of these other named networks is not supported on SR Linux. When the network is defined as **host**, SR Linux starts the container in the network namespace in which the StartContainer RPC is received.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- **cap**: The capabilities granted to or revoked from the container. Each capability is defined as **add** or **remove**. Supported capabilities are: **AUDIT_WRITE**, **CHOWN**, **DAC_OVERRIDE**, **FOWNER**, **FSETID**, **KILL**, **MKNOD**, **NET_BIND_SERVICE**, **NET_RAW**, **SETFCAP**, **SETGID**, **SETPCAP**, **SETUID**, and **SYS_CHROOT**
- **restart**: The restart policy for the container. The available options are: **NONE**, **ALWAYS**, **UNLESS_STOPPED**, and **ON_FAILURE**. Nokia recommends setting containers to the **ALWAYS** restart policy to ensure that they are restarted upon boot and CPM switchover. The number of restrarts that a container attempts before being declared as failed can be specified in the **attempts** field of the **Restart** message.
- **run_as**: The user the container runs as. If set, SR Linux starts the container using the `user` or `user:group` notation..
- **labels**: The labels assigned to a container.
- **limits**: The resource limitation requirements for the container: **max_cpu**, **soft_mem_bytes**, and **hard_mem_bytes**. The **max_cpu** provides the maximum amount of CPUs that may be consumed (for example 0.5 means a maximum of half a CPU). It does not imply this CPU quantity is reserved for this container. The hard and soft memory limits (in bytes) define the memory consumption levels at which the container may be terminated (soft) or is terminated (hard).
- **devices**: The devices added to the container. Permissions for the specified device are derived from the provided permissions enum: **r** = READ, **w** = WRITE, **m** = MKNOD.
- **location**: The CPM onto which the container is scheduled. The supported options are **L_UNKNOWN**, **L_PRIMARY**, **L_BACKUP** and **L_ALL**. The primary CPM is the currently active CPM. If a CPM switchover occurs, then the primary and backup CPMs are switched. For example, a container scheduled as **L_PRIMARY** is placed onto the active CPM A then a CPM switchover occurs and the container is then scheduled onto CPM B as the **L_PRIMARY** CPM is now the one in slot B.

  Containers scheduled as **L_UNKNOWN** are placed onto the active CPM. Containers scheduled as **L_ALL** are started on both the active and standby CPMs. These are independent containers with independent storage. The containers are started in the same network namespace and so may be able to communicate with one another, depending on the selected options. Containers do not support high-availablilty switchover. In the case of a CPM switchover they are essentially destroyed and recreated.

StartContainer RPC structure

```
rpc StartContainer(StartContainerRequest) returns (StartContainerResponse) {}
```

8.9.6.1 Custom container location placement

When a container is started on a dual-CPM chassis, the Containerz **Location** field specifies which CPMs to run the container on: the active CPM, the standby CPM, or both CPMs.

To start a container, the StartContainer RPC uses a StartContainerRequest message. Within this message, the **Location** field supports the following values:

- **L_PRIMARY** — Starts the container on the active CPM.
- **L_BACKUP** — Starts the container on the standby CPM.
- **L_ALL** — Starts the container on both CPMs, if available.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

If the container cannot start on one of the CPMs, that is not an error. For example, in a single CPM device, only one copy of the container is started. Containers started with this value start in the same (management) network instance.

- **L_UNKNOWN** — Starts the container on the active CPM when the location is unknown.

The active CPM is not always CPM A and the standby is not always CPM B. The router determines which CPM is active and which is standby and starts the containers accordingly.

The system retains the requested start location. For example, if the container is started using **L_ALL** on a dual-CPM system, and the standby CPM is unavailable at the time of the request, the container starts only on the active CPM. However, if the standby CPM subsequently comes back online, the system attempts to start the container on the standby CPM, which may or may not succeed. None of these scenarios constitute an error.

Labels

The following labels are applied to each container:

- **net.openconfig.containerz.location.intended** — Reports the location in the initial request: **L_UNKNOWN** , **L_PRIMARY**, **L_BACKUP**, or **L_ALL**.
- **net.openconfig.containerz.location.actual** — Reports the actual location where that instance of the container started: **L_PRIMARY** or **L_BACKUP**. For example, if the container started on the currently active CPM the value is **L_PRIMARY**. If the container is not successfully started, a label may not be applied.
- **net.openconfig.containerz.location.reason** — Displays a reason when a container fails to start on a specific CPM; taken from the Docker failed-to-start reason.

Docker daemon

Only one Docker daemon runs on each CPM, running in the management network instance. All gNOI requests to start containers enter in this namespace.

The **Network** Containerz option defines the type of networking that is made available to the container: **host** , **bridged**, or any other explicit network type available on the router. If **host** is selected, the container is started in the network namespace where the Docker daemon is running. If **bridged** is selected, this is accepted on SR Linux, but no action is taken.

The Docker engine runs independently on each CPM and the processes do not communicate.

Container communication

Containers running on different CPMs can communicate with one another. Containers running on the primary CPM can also communicate with external systems, including management systems and containers running on other network nodes. Containers running on the standby CPM do not have external connectivity; however, they can communicate with containers in the same (management) network instance on the primary and standby CPMs.

CPM failure and switchover

If the active CPM fails and the standby CPM becomes active, containers placed on the **L_PRIMARY** CPM are migrated to the new active CPM. For example:

- Containerz starts a container with **Location** = **L_PRIMARY** and it ends up on the active CPM A. If CPM A fails, CPM B takes over as active. The container is then rescheduled and started on CPM B but is still marked in the label as being on the primary CPM.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- Containerz starts Container 1 with **Location** = **L_PRIMARY** and Container 2 with **Location** = **L_BACKUP**. CPM A is the active CPM and CPM B is the standby CPM. In this case, Container 1 is placed onto CPM A with the **net.openconfig.containerz.location.actual** set to **primary** and the **net.openconfig.containerz.location.intended** label set to **L_PRIMARY** . Container 2 is placed onto CPM B with the **net.openconfig.containerz.location.actual** set to **backup** and the **net.openconfig.containerz.location.intended** label set to **L_BACKUP**.

  If a CPM switchover is initiated, CPM B becomes the active CPM and CPM A becomes the standby CPM. In this case, the primary container, Container 1, migrates to CPM B with its labels unchanged and the backup container, Container 2, migrates to CPM A, also with its labels unchanged. Each container is rescheduled on the CPM that matches its intended role (L_PRIMARY on the new active, L_BACKUP on the new standby). L_ALL remains on both CPMs (where possible). As the containers are rescheduled, they are removed and then redeployed.

8.9.7 StopContainer RPC

The StopContainer RPC stops the container specified by the **instance_name** in the StopContainerRequest message on the target node. The container is gracefully stopped unless the **force** option is set to **True**. In addition, if the **restart** option is set to **True**, the container is restarted after it has stopped.

StopContainer RPC structure

```
rpc StopContainer(StopContainerRequest) returns (StopContainerResponse) {}
```

8.9.8 UpdateContainer RPC

The UpdateContainer RPC updates the specified container on the target node to the image specified in the UpdateContainerRequest message.

The UpdateContainer RPC is by default, synchronous, meaning that it performs the upgrade and only returns after the activity is complete. To request an asynchronous update, set **async** to **True** in the UpdateContainerRequest message. In this case, SR Linux performs the admission checks to ensure that the request can proceed (such as whether the image exists) before completing the RPC. You must then monitor the upgrade process externally.

In the UpdateContainerRequest message, specify the **instance_name**, **image_name**, and **image_tag**.

A container update may fail for a number of reasons, including: referencing a non-existent container, an image or tag cannot be found, the required ports cannot be exposed, or the container is found but is not in a started state.

UpdateContainer RPC structure

```
rpc UpdateContainer(UpdateContainerRequest) returns (UpdateContainerResponse) {}
```

8.9.9 Log RPC

The Log RPC streams the logs from the specified container. If the container is not running, the RPC returns the latest available logs. The instance_name is mandatory when calling the Log RPC.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Log RPC structure

```
rpc Log(LogRequest) returns (stream LogResponse) {}
```

8.9.10 CreateVolume RPC

The CreateVolume RPC creates the specified volume on the target node. The volumes are persistent and remain available after a reboot.

The CreateVolumeRequest accepts the following options:

- **name**: The name of the volume. If omitted, SR Linux creates one automatically
- **driver**: The filesystem driver associated with the volume. The following options are supported:
  - **DS_LOCAL**: Not specifying a driver results in the **DS_LOCAL** option being selected.
  - **DS_CUSTOM**: SR Linux supports only one instance of **DS_CUSTOM**; that is, the system cannot support two CreateVolumeRequests with driver type **DS_CUSTOM** and any other options.
- **options**: Two types of options are supported:
  - **local_mount_options**: The local filesystem mount options supported are: **type**, **options** and **mountpoint**. SR Linux supports the **none type** only. The options are joined into the Docker DriverOpts (`o`). If a propagation mode is not specified SR Linux appends `,rprivate` to avoid inheriting host mount propagation which may impact operational performance. The mountpoint is a bind-style local mount.
  - **custom_options**: A set of key, value pairs that are appended to Docker DriverOpts verbatim.

There are no restrictions on the SR Linux filesystem path that may be selected as the location of a volume. Select the location with caution. If used incorrectly, it may render the system inoperable.

CreateVolume RPC structure

```
rpc CreateVolume(CreateVolumeRequest) returns (CreateVolumeResponse) {}
```

8.9.11 RemoveVolume RPC

The RemoveVolume RPC removes the specified volume from the target node. If the specified volume does not exist, the RPC completes with no action taken.

RemoveVolume RPC structure

```
rpc RemoveVolume(RemoveVolumeRequest) returns (RemoveVolumeResponse) {}
```

8.9.12 ListVolume RPC

The ListVolume RPC returns all volumes from the target node that match the filter specified in the ListVolumeRequest message.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

ListVolume RPC structure

```
rpc ListVolume(ListVolumeRequest) returns (stream ListVolumeResponse) {}
```

8.9.13 StartPlugin RPC

The StartPlugin RPC starts the specified plugin on the target node.

StartPlugin RPC structure

```
 rpc StartPlugin(StartPluginRequest) returns (StartPluginResponse) {}
```

8.9.14 StopPlugin RPC

The StopPlugin RPC starts the specified plugin on the target node.

StopPlugin RPC structure

```
rpc StopPlugin(StopPluginRequest) returns (StopPluginResponse) {}
```

8.9.15 ListPlugin RPC

The ListPlugin RPC returns all containers from the target node that match the filter specified in the ListPluginRequest message.

ListPlugin RPC structure

```
rpc ListPlugins(ListPluginsRequest) returns (ListPluginsResponse) {}
```

8.9.16 RemovePlugin RPC

The RemovePlugin RPC removes the specified plugin from the target node. However, the plugin is not removed if it is still enabled. You must first call the StopPlugin RPC before attempting to invoke the RemovePlugin RPC.

RemovePlugin RPC structure

```
rpc RemovePlugin(RemovePluginRequest) returns (RemovePluginResponse) {}
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

8.10 Container resource management using hierarchical cgroups

Note: Hierarchical cgroups for Containerz containers are supported on the 7250 IXR Gen 2c+ and 7250 IXR Gen 3 platforms.

Hierarchical cgroups can provide granular resource management for containers that are managed by the gNOI Containerz service. All Containerz containers are organized in a unified hierarchy under the same cgroup parent (`/workload.slice/secondary/containerz`). This hierarchy allows the system to enforce aggregate resource limits for CPU and memory across all managed containers.

SR Linux follows cgroup v2 hierarchical rules. For example, even if a container's **memory-max** is set to **0** (no limit), it is still strictly limited by the parent cgroup's defined memory maximum.

Parent container

Typically, Docker uses `/docker` (`cgroupfs`) or `/system.slice` (`systemd`) for the parent container. SR Linux overrides the default to use `/workload.slice/secondary/containerz` as the parent instead. To prevent containers from being nested under `/system.slice` by the Docker daemon, SR Linux uses a leading forward slash (`/`) in the CgroupParent configuration. This forces the path to be relative to the cgroup root.

Default Containerz cgroup values

The default SR Linux cgroup profile settings include the following:

- cgroup path: `workload.slice/secondary/containerz`
- memory max: 0.5 (50% of available RAM)
- CPU weight: 100
- CPU quota: 0 (no limit)

8.10.1 Configuring CPU and memory limits using cgroups

Procedure

To modify the cgroup limits, use the following commands.

- **platform control cgroup memory-controller memory-max**
- **platform control cgroup cpu-controller cpu-weight**
- **platform control cgroup cpu-controller cpu-max**

Each CPM is configured by referencing the control module identifier (**control A** or **control B**).

Example: Modify cgroup limits

```
-{ candidate shared default }--[  ]--
# info with-context platform control A cgroup workload.slice/secondary/containerz
    platform {
        control A {
            cgroup workload.slice/secondary/containerz {
                memory-controller {
                    memory-max 26808180736
                }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
                cpu-controller {
                    cpu-weight 99
                    cpu-max 100000
                }
            }
        }
    }
```

8.10.2 Verifying the cgroup configuration

About this task

To validate that the container is correctly placed and limited, perform the following steps.

Procedure

Step 1. Launch the container.

Step 2. Use the gNOI Containerz ListContainer RPC to obtain the container ID value.

Step 3. Use the **tools system cgroup command pgrep cgroup workload.slice/secondary/containerz/**<*container-ID*> command to obtain the PID associated with the container.

```
--{ candidate shared default }--[  ]--
# tools system cgroup command pgrep cgroup workload.slice/secondary/containerz/
<container-id>

+--------+---------+
|  Pid   | Process |
+========+=========+
| 251591 | sleep   |
+--------+---------+
```

Step 4. To verify the cgroup membership, inspect the cgroup for the container's PID using the **cat /proc/**<*PID*>**/cgroup** bash command:

```
--{ candidate shared default }--[  ]--
# bash cat /proc/251591/cgroup
# Expected: 0::/workload.slice/secondary/containerz/<container_id>
```

Step 5. Verify the effective limit is applied at the parent level:

```
--{ candidate shared default }--[  ]--
# bash cat /sys/fs/cgroup/workload.slice/secondary/containerz/memory.max
```

Step 6. To verify the resource limit configuration, modify **cpu-max**, **cpu-weight**, and **memory-max** and confirm that the values are propagating correctly from the CLI to the cgroup file system.

```
--{ candidate shared default }--[  ]--
# bash
:/# cat /sys/fs/cgroup/workload.slice/secondary/containerz/cpu.weight
99
:/# cat /sys/fs/cgroup/workload.slice/secondary/containerz/memory.max
26808180736
:/# cat /sys/fs/cgroup/workload.slice/secondary/containerz/cpu.max
max 100000
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

8.11 gRPC YANGAction service for YANG-modeled actions

SR Linux supports the gRPC YANGAction service to execute YANG-modeled actions over gRPC.

Note: YANGAction is a Nokia-defined gRPC service, distinct from OpenConfig's standardized gNOI services.

To support vendor-specific operational commands using gRPC, SR Linux supports YANG-modeled actions for operational commands using YANG **action** statements (see YANG-modeled operations). The YANGAction service complements gNOI, providing a generic transport for YANG **action** statements. It enables vendor-specific operations to be exposed over a standardized gRPC interface without requiring a new proto definition each time a new YANG-modeled action is added. For more information about supported YANG-modeled actions, see YANG model of an action.

Path encoding

Input and output data for the YANGAction service is encoded using JSON-IETF (RFC 7951). The path to a YANG action node is encoded using the gNOI **types.Path** type, consistent with other gNOI services, and points to the YANG **action** node itself. Per RFC 7950 section 7.15, the **input** and **output** containers of the action are intentionally omitted from the path.

For example, the path for `/network-instance[name=default]/protocols/bgp/neighbor[peer-address=1.1.1.1]/reset-peer` is encoded as follows:

```
{
"elem": [
    { "name": "network-instance", "key": { "name": "default" } },
    { "name": "protocols" },
    { "name": "bgp" },
    { "name": "neighbor", "key": { "peer-address": "1.1.1.1" } },
    { "name": "reset-peer" }
  ]
}
```

ExecAction RPC

The YANGAction service exposes a single RPC, ExecAction, which accepts an ActionRequest and returns a stream of ActionResponse messages. The ActionResponse carries a single field: the JSON-IETF encoded output. The gRPC server streams ActionResponse messages as output becomes available, allowing long-running actions (such as ping) to return results progressively without buffering.

YANGAction proto

The proto file for the YANGAction service is available at the following location: https://github.com/nokia-steward/srlinux-protobufs. Fields marked **reserved** for v2 are present in the schema for forward compatibility but are not populated by any v1 server implementation.

8.11.1 YANG action example: BGP neighbor hard reset

The following example illustrates a simple YANG action case. The BGP neighbor hard reset action takes no input and returns no structured output. The stream closes immediately with RPC status **OK** on success. No ActionResponse messages are sent.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: YANG action definition (simplified)

The following output shows the BGP neighbor hard reset action.

```
list neighbor {
  key peer-address;
  leaf peer-address { type srl-comm:ip-address; }

  action reset-peer {
    // No input container.
    // No output container.
  }
}
```

Example: SR Linux CLI equivalent

The equivalent CLI command for the neighbor reset action is as follows.

```
# tools network-instance default protocols bgp neighbor 10.1.1.1 reset-peer
```

Example: gRPC ActionRequest

The gRPC ActionRequest for the neighbor reset action is as follows.

```
{
  "action_path": {
    "elem": [
      { "name": "network-instance", "key": { "name": "default" } },
      { "name": "protocols" },
      { "name": "bgp" },
      { "name": "neighbor", "key": { "peer-address": "10.1.1.1" } },
      { "name": "reset-peer" }
    ]
  },
  "input_json_ietf": ""
}
```

Example: gRPC stream behavior — success

If the action succeeds, no ActionResponse messages are sent. The stream closes with RPC status OK.

Example: gRPC stream behavior — invalid input value

If the action fails, no ActionResponse messages are sent. The stream closes as follows.

```
RPC status: NOT_FOUND
Message:    "neighbor 10.1.1.1 not found in network-instance default"
```

8.12 gNOI configuration

SR Linux supports gNOI services using the gRPC server configuration. To enable gNOI support, enable the gRPC server.

The session between the gNOI client and SR Linux must be encrypted using TLS.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

See the "Management servers" chapter in the *SR Linux Configuration Basics Guide* for information about how to configure the gRPC server.

8.12.1 Configuring gNOI services

About this task

As part of the gRPC server configuration, you can also specify which individual gNOI services to enable.

Procedure

To enable gNOI services, use the **system grpc-server** <*network-instance*> **services** command.

Example: Enable gNOI services

```
# info with-context system grpc-server mgmt
    system {
        grpc-server mgmt {
            admin-state enable
            timeout 7200
            rate-limit 1500
            session-limit 20
            metadata-authentication true
            yang-models native
            tls-profile tls-profile-1
            network-instance mgmt
            port 50052
            services [
                gnoi.packet_link_qualification
            ]
            source-address [
                ::
            ]
            gnmi {
                commit-confirmed-timeout 0
                commit-save false
                include-defaults-in-config-only-responses false
            }
            unix-socket {
                admin-state disable
            }
        }
    }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

9 gNSI

gRPC Network Security Interface (gNSI) is a set of RPCs that allow a client to define the security configuration of a device and to retrieve security information. gNSI is maintained by the OpenConfig project on Github, and contains a set of RPCs as well as extensions to gNMI and the OpenConfig YANG modules.

SR Linux supports the following gNSI services:

- gNSI Authz service
- gNSI Certz service

9.1 gNSI Authz service

The gNSI Authz service allows clients to configure and manage a gRPC-level authorization policy that defines which users can access which gRPCs on the target node. This policy is defined by a JSON string and controls access to all gRPC servers to prevent access from malicious actors.

Note: Only one policy is active for the entire system.

Unlike role-based access control, which provides authorization of system YANG paths, the Authz service authorizes the use of individual gRPCs; for example access to the `/gnmi.gNMI/Get` RPC, rather than to any individual path that the Get RPC can retrieve.

Typical Authz service steps

In the typical Authz service process, the client performs the following steps:

1. Upload the authorization policy by sending an UploadRequest to the target node using the Rotate RPC.
2. Verify that the authorized users can access the required gRPCs (and that non-authorized users are denied) by sending a ProbeRequest using the Probe RPC.
3. Lock in the updates by sending a FinalizeRequest using the Rotate RPC.
4. As required, retrieve details of the currently active policy by sending a GetRequest using the Get RPC.

9.1.1 Authz policy

The Authz policy consists of the following:

- **name**: name for the policy defined by the client
- **allow_rules list** (mandatory): list of allow rules for the policy
- **deny_rules list** (optional): list of deny rules for the policy

Allow and deny rules

Each allow or deny list is comprised of instances of rules. Each rule consists of the following:

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- **name**: name for the rule defined by the client; similar to the policy name
- **source**: a holding container with a single source type of **principals**:
  - **principals**: a list of principals that must match against user objects, be it one of the users defined in **system aaa authentication user**, a remotely resolved user from a **system aaa server-group**, or the **admin**
- **request** contains:
  - **paths**: a list of gRPC RPC paths (in format `pkg.Service/Rpc`) that this policy applies to; for example: `/gnoi.os.OS/Install` (if no paths are provided, all paths are matched)
  - **headers**: a list of key/value pairs matching HTTP headers

Note: SR Linux does not support matching on **headers**.

Matching source and request

Both the source and request fields must match in order for a rule to match. If both source and request are empty, the rule always matches. Empty source and request can serve as a wildcard in a **deny_rules** list to deny all RPCs or in an **allow_list** to accept all RPCs except those matching in the **deny_rules** list.

If only the source is empty, the paths provided in the request field apply to all users system wide. Similarly if only the request is empty then the action (deny or allow) is applied to all users provided in the source. If no paths are specified in the request, all paths are matched.

Example: Authz policy example

The following shows an example policy in which two administrators Alice and Bob have the ability to execute any of the gNSI Authz RPCs.

```
{
  "name": "gNSI.authz policy",
  "allow_rules": [{
    "name": "admin-access",
    "source": {
     "principals": [
      "spiffe://nokia.com/sa/alice",
      "spiffe://nokia.com/sa/bob"
      ]
    },
    "request": {
      "paths": [
        "/gnsi.authz.Authz/*"
      ]
    }
  }]
}
```

Authz policy persistence

As it is a requirement to persist a policy after it has been finalized, on finalization the current policy is written to `/etc/opt/srlinux/gnsi/authz-policy.json`. This policy is read by the SR Linux gNSI server on startup and exposed as the active policy.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

It is an Authz requirement that if a policy is configured, no gRPC servers (this includes P4RT, gRIBI, gNOI, gNMI, and gNSI externally) can start until the policy is successfully activated. The Authz scope is to protect access to system RPCs from malicious entities, so a policy must be active before ports are opened.

Default Authz policy

As part of the default SR Linux configuration, the Authz service is running, and a default Authz policy controls access to every active gRPC service. The result of the default policy is that any *authenticated* user is permitted full access to any currently active service and gRPC. This does not overrule any subsequent role-based access control that exists on the system.

```
{
 "name": "Default policy",
 "deny_rules": [],
 "allow_rules": [
  {
   "name": "admin-access",
   "source": {
    "principals": [
     "*"
    ]
   },
   "request": {
    "paths": [
     "/*",
     ""
    ],
    "headers": []
   }
  }
 ]
}
```

9.1.2 Authz authorization logic and interactions

To validate a gRPC authorization, gNSI performs the following actions:

1. Check the user and RPC combination against any **deny_rules**. If a match is found, the request is denied. If no match is found or there are no **deny_rules**, execute the next step.
2. Check for a match against any **allow_rules**. If a match is found, the request is permitted. If no match is found, deny the request.

Note: The gNSI Authz behavior differs from role-based access control, in that, if any **deny_rules** are matched, then the request is denied, even if a more specific path exists in **allow_rules**.

Authz interactions with role-based access control and service authorization

The Authz service operates in addition to existing role-based access control and service authorization. As a result, to provide a user access to a gRPC, you must configure their access under role-based access control and service authorization (for example, **system aaa authorization role services gnmi**) in addition to enabling access using the Authz service. The **system aaa authorization role services** command also supports defining service authorization for **gnsi**.

Order of authorization

The order of authorization is as follows:

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

1. Check service authorization first; only if the user is authorized for the service are they admitted.
2. If the interface is a gRPC interface, check the Authz policy to see if the user is authorized for the specified RPC.
3. If the user is configuring or reading any paths, perform role-based access control.

9.1.3 Default TLS profile

To allow the gNMI (and gNSI) server to start as part of the default configuration, the factory default configuration includes a TLS profile named **__default__**. This default policy does not authenticate the identity of connecting clients. You can view the policy using the **info from state system tls server-profile __default__** command.

```
# info with-context from state system tls server-profile __default__
   system {
        tls {
            server-profile __default__ {
                key $aes1$4lL6SVq5G8=$FqANWbk8TQmrGYB1cTuaplFUZTZlAP992Debrq4X2ZjfOde4wA
                ...
                certificate "-----BEGIN CERTIFICATE-----
MIIFLzCCAxegAwIBAgIUBQFYPrWyBalNUfTNCBUAMiWrk2kwDQYJKoZIhvcNAQEL
BQAwHTEOMAwGA1UEAwwFc3JsZDUxCzAJBgNVBAYTAlVTMCAXDTI0MDIyMDIzNTY0
...
-----END CERTIFICATE-----
"
                authenticate-client false
                dynamic true
                cipher-list [
                    ecdhe-ecdsa-aes256-gcm-sha384
                    ecdhe-ecdsa-aes128-gcm-sha256
                    ecdhe-rsa-aes256-gcm-sha384
                    ecdhe-rsa-aes128-gcm-sha256
                ]
            }
        }
    }
```

SR Linux also supports a default TLS configuration option for the gRPC server (**system grpc-server** *name* **default-tls-profile [true | false]**) that controls whether the system uses the default TLS profile if none are configured using the **tls-profile** option. If **default-tls-profile** is set to **false** (the default value), and no **tls-profile** is defined, then TLS is disabled entirely from the system.

9.1.4 Supported RPCs

SR Linux supports the following Authz service RPCs:

- Rotate RPC
- Probe RPC
- Get RPC

9.1.5 Rotate RPC

The Authz Rotate RPC allows a client to replace an existing gRPC authorization policy on the target node.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

If any steps in the Rotate RPC process fail, the target node rolls back any changes made by the RPC.

Only one Rotate RPC can be in progress at a time. If a client attempts to call the RPC while another call is already in progress, the second call is rejected with a gRPC error of `UNAVAILABLE`.

Rotate RPC structure

```
rpc Rotate(stream RotateAuthzRequest) returns (stream RotateAuthzResponse);
```

9.1.6 Probe RPC

The Authz Probe RPC tests the currently active policy. It allows the Authz policy engine on the target node to provide a response to a specified gRPC call without actually performing the gRPC operation. The current policy can be one that has not been finalized, but this is not a requirement.

The Probe RPC expects a single ProbeRequest message, and returns a ProbeResponse. The ProbeRequest does not result in an actual gRPC call being made; it simply validates whether the call can succeed with the provided user and RPC information.

Probe RPC structure

```
rpc Probe(ProbeRequest) returns (ProbeResponse);
```

9.1.7 Get RPC

The Authz Get RPC returns the currently active gRPC authorization policy, as well as the policy version and created-on timestamp.

Get RPC structure

```
rpc Get(GetRequest) returns (GetResponse);
```

9.1.8 gNSI Authz tools and state commands

To allow the gNSI Authz behavior to be invoked in the CLI or by a gNMI or JSON-RPC client, SR Linux supports the following tools commands:

- **tools system aaa authorization authz-policy clear**
- **tools system aaa authorization authz-policy probe**
- **tools system aaa authorization authz-policy remove**
- **tools system aaa authorization authz-policy rotate**

Note: Performing a **rotate** operation using the tools commands triggers an immediate finalization.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

9.1.8.1 Clearing Authz policy counters using the tools command

Procedure

To clear Authz policy counters, use the **tools system aaa authorization authz-policy clear** command.

Example: Clear Authz policy counters

```
--{ candidate shared default }--[  ]--
# tools system aaa authorization authz-policy clear
```

9.1.8.2 Probing Authz policy counters using the tools command

Procedure

To probe an Authz policy, use the **tools system aaa authorization authz-policy probe** command. The **rpc** parameter specifies which RPC to test, and the **user** parameter specifies which user to use for the test.

Example: Probe Authz policy counters

```
--{ candidate shared default }--[  ]--
# /tools system aaa authorization authz-policy probe rpc /gnmi.gNMI/Get user admin
/system/aaa:
    Authz authorization policy version '1' action for user 'admin' and rpc '/gnmi.gNMI/
Get' is ACTION_PERMIT
```

9.1.8.3 Removing an Authz policy using the tools command

Procedure

To remove an Authz policy, use the **tools system aaa authorization authz-policy remove** command.

Given that there is only one system-wide gRPC authorization policy, this command reverts to the factory default authorization policy, which authorizes any gRPC call for every authenticate user.

Example: Remove Authz policy

```
--{ candidate shared default }--[  ]--
# tools system aaa authorization authz-policy remove
```

9.1.8.4 Rotating an Authz policy using the tools command

Procedure

To rotate an Authz policy, use the **tools system aaa authorization authz-policy rotate** command, which supports the following parameters:

- **created-on**: sets the created-on value for the new policy
- **policy**: specifies the gRPC authorization policy as a JSON-formatted string
- **version**: specifies a version string to store with the policy

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Unlike the Rotate RPC, the **tools system aaa authorization authz-policy rotate** command triggers an immediate finalization.

Example: Rotate Authz policy

```
--{ candidate shared default }--[  ]--
# / tools system aaa authorization authz-policy rotate policy "{\"name\": \"foo\", \
"allow_rules\": { \"name\": \"dev\", \"source\": { \"principals\": [
\"admin\", \"foo\", \"bar\" ] }, \"request\": { \"pat hs\": [ \"/*\", \"\" ] } } }"
 created-on 10 version 1
/system/aaa:
    Authz authorization policy has been rotated and finalized (version '' created on
 '1970-01-01T00:00:10.000Z')
```

9.1.8.5 Displaying the currently installed Authz policy

Procedure

To display the currently installed Authz policy, use the **info from state system aaa authorization authz-policy** command.

Example: Display the currently installed Authz policy

```
# info with-context from state system aaa authorization authz-policy
    system {
        aaa {
            authorization {
                authz-policy {
                    version 2023-06-01
                    created-on "2024-02-16T17:29:08.721Z (40 minutes ago)"
                    policy "{
 \"name\": \"Default policy\",
 \"deny_rules\": [],
 \"allow_rules\": [
  {
   \"name\": \"admin-access\",
   \"source\": {
    \"principals\": [
     \"*\"
    ]
   },
   \"request\": {
    \"paths\": [
     \"/*\",
     \"\"
    ],
    \"headers\": []
   }
  }
 ]
}
"
                }
            }
        }
    }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

9.2 gNSI Certz service

The gNSI Certz service allows a client to replace a certificate, trust bundle, certificate revocation list (CRL), or some combination of these artifacts on a target node.

Typical Certz service steps

In the typical certificate rotation process, the client performs the following steps using the Rotate RPC:

1. Either generate the CSR on the client side, or direct the target node to generate the CSR using a GenerateCSRRequest.
2. Obtain a signature for the certificate from the CA (out of band of the RPC).
3. Upload the signed certificate to the target node using an UploadRequest.
4. Verify (out of band of the RPC) that the services that use the new certificate bundle continue to operate normally.
5. Send a FinalizeRequest to the target node to lock in the updates.

Entity messages

The Certz service uses Entity messages to define the Certz artifacts, which can be any of the following:

- **Certificate**: An X.509 certificate and optional private key that is either PEM or DER encoded. The specific encoding is indicated in fields carried in children of the Entity message. A certificate also specifies any intermediate parent certificates. It is equivalent to the certificate displayed using the **system tls server-profile** *name* **certificate** command.
- **Trust bundle**: The certificate bundle used to validate outbound connections, and inbound clients when mTLS is enabled. This is a typical system trust bundle available in `/etc/ssl/certs`. It is equivalent to the trust anchor displayed using the **system tls server-profile** *name* **trust-anchor** command.
- **CRL**: The CRL bundle, which lists certificates that have been revoked and can no longer be used to validate connections. It is equivalent to the CRL displayed using the **system tls server-profile** *name* **certificate-revocation-list** command.

All together, these artifacts make up a single profile. In a single Rotate RPC call, a client can rotate all artifacts, or only specific artifacts.

To improve performance on the target node, certificates can be ordered. Groups of chained certificates (namely the **trust-anchor** and **crl**) appear last, and within the group, the root certificate must be the last certificate in the chain; for example: CertA, CertB, CertB-Root, CertC, CertC-Intermediate, CertC-Root.

Note: SR Linux can support an unordered bundle.

Default TLS profile

The default TLS profile is available at system startup and uses system-generated self-signed certificates and private keys. The default profile is saved to disk with other TLS profiles at `/etc/opt/srlinux/tls`. This default profile allows the gRPC server to start as part of the default configuration to meet the requirements of the Authz service.

TLS profiles can be populated using the Certz service or configured via the CLI using the **system tls server-profile** command. If a conflict exists in the configuration, the CLI configuration takes precedence.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Certz state paths

Certz state paths are available under **system tls server-profile certz**. These paths indicate the certificate, trust bundle, and CRL in use by each specific instance of a gRPC server.

The Certz JSON files are stored in `/etc/opt/srlinux/gnsi`, while the CRL PEM files are stored in `/etc/opt/srlinux/tls`.

Only users in the **tls** group can read these artifacts, and only the **srlinux** user can modify them.

Supported RPCs

SR Linux supports the following Certz service RPCs:

- Rotate RPC
- CanGenerateCSR RPC
- AddProfile RPC
- DeleteProfile RPC
- GetProfileList RPC

9.2.1 Rotate RPC

The Certz Rotate RPC allows a client to replace an existing certificate, trust bundle, CRL, or some combination of these artifacts on the target node. Either the target node or the client can generate the CSR for the new device certificate. If the client generates the CSR, it must also provide the corresponding private key with the signed certificate.

Only one Rotate RPC can be active at once. If SR Linux detects an additional Rotate RPC, it returns an error.

The Rotate RPC accepts a stream of RotateCertificateRequest messages and returns a stream of RotateCertificateResponse messages in response. A client uses the RPC to upload a certificate by specifying the certificate in a **rotate_request** before testing the new certificate (out of band of the RPC) and finalizing the result with a **finalize_rotation**. Until the target node receives the **finalize_rotation**, the change is transient and not persistent on disk. If the RPC is canceled for any reason before a **finalize_rotation** is received, then the system reverts all services using TLS to the previous values.

If the stream is broken or any of the steps fail, the target node rolls back to the original state, reverting any changes to any artifact.

Rotate RPC structure

```
rpc Rotate(stream RotateCertificateRequest) returns (stream RotateCertificateResponse);
```

9.2.1.1 Rotate RPC use cases

The following sections describe a number of Rotate RPC use cases each presenting the expected sequence of message exchange.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Case 1: Client generates the CSR and gets it signed

1. The Rotate RPC stream begins.

```
Client <------          Rotate() RPC stream begins          ------> Target node
```

2. The client generates the CSR and gets the certificate signed by the CA (typically done before initiating the stream).
3. The client sends the signed certificate with private key to the target node.

   The client can optionally include a trust bundle, a CRL, or both.

```
Client ------>  UploadRequest(certificate, [trust_bundle],  ------> Target node
                        [certificate_revocation_list])

Client <------                UploadResponse                <------ Target node
```

4. To validate that the certificate works, the client attempts to create new connections to the target node using the new certificate. If the new connections fail, the client cancels the RPC, forcing the target node to roll back all certificates included in the RPC.

   If a new CRL bundle is included in the UploadRequest in step 3, the client also attempts to establish new connections to the target node using the revoked certificates. In this case, failed connections validate that the certificates have been revoked.
5. After successful validation, the client sends a final commit.

```
Client ------>                FinalizeRequest               ------> Target node
```

Case 2: Target node generates the CSR and the client gets it signed

1. Rotate RPC stream begins:

```
Client <------           Rotate() RPC stream begin          ------> Target node
```

2. The client sends a GenerateCSRRequest to the target node, and the target node provides the CSR in response:

```
Client ------>              GenerateCSRRequest              ------> Target node

Client <------              GenerateCSRResponse             <------ Target node
```

3. The client obtains a signature for the certificate from the CA.
4. The client sends the signed certificate without private key to the target node (the target node already has the private key because it generated the CSR).

   The client can optionally include a trust bundle, a CRL, or both.

```
Client ------>  UploadRequest(certificate, [trust_bundle],  ------> Target node
                      [certificate_revocation_list])

Client <------                UploadResponse                <------ Target node
```

5. The client tests and validates the certificate, using the same validation step as in Case 1.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

6. After successful validation, the client sends a final commit.

```
Client ------>                FinalizeRequest             ------> Target node
```

Case 3: Client changes only the trust bundle

1. The Rotate RPC stream begins:

```
Client <------             Rotate() RPC stream begin       ------> Target node
```

2. The client sends a Certificate Authority Bundle chain to the target node.

   The client can optionally include a CRL.

```
Client ------>          UploadRequest(trust_bundle,         ------> Target node
                       [certificate_revocation_list])

Client <------                  UploadResponse              <------ Target node
```

3. The client tests and validates the certificate, using the same validation step as in Case 1.
4. After successful validation, the client sends a final commit.

```
Client ------>                  FinalizeRequest             ------> Target node
```

Case 4: Client uploads a CRL

1. The Rotate RPC stream begins:

```
Client <------           Rotate() RPC stream begin           ------> Target node
```

2. The client sends a CRL bundle to the target node.

```
Client ------>  UploadRequest([certificate_revocation_list])   ------> Target node

Client <------                UploadResponse                  <------ Target node
```

3. The client validates the CRL by attempting to establish a new connection to the target node using the revoked certificates. Failed connections validate that the certificates have been revoked.
4. After successful validation, the client sends a final commit.

```
Client ------>                 FinalizeRequest                ------> Target node
```

9.2.2 CanGenerateCSR RPC

The Certz CanGenerateCSR RPC tests whether the target node is able to generate a CSR given provided parameters. The CanGenerateCSR RPC consists of a single CanGenerateCSRRequest message and returns a CanGenerateCSRResponse. The client includes the parameters that it wants the target node to test for the CSR generation, including:

- **key_type**: the algorithm used for generation of the key pair (for example, **KEY_TYPE_RSA**)
- **signature_algorithm_type**: the signature algorithm used to generate the key pair (for example: **SIGNATURE_ALGORITHM_SHA512_WITH_RSA**)

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- **certificate_type**: the type of certificate to create (for example, **CERTIFICATE_TYPE_X509**)
- **key_size_bits**: a uint32, indicating the size of the key in bits (for example, **2048**)

CanGenerateCSR RPC structure

```
rpc CanGenerateCSR(CanGenerateCSRRequest) returns (CanGenerateCSRResponse);
```

9.2.3 AddProfile RPC

The Certz AddProfile RPC allows a client to add a new TLS profile to the target node. All elements of the added profile (certificate, CA trust bundle, and CRLs) are initially empty. The client must subsequently populate the artifacts using a Rotate RPC.

If a client attempts to add a pre-existing profile, the attempt is rejected with an error.

AddProfile RPC structure

```
rpc AddProfile(AddProfileRequest) returns (AddProfileResponse);
```

9.2.4 DeleteProfile RPC

The Certz DeleteProfile allows a client to remove an existing TLS profile.

The **__default__** profile used by the gRPC server cannot be deleted. If a client attempts to delete the **__default__** profile, the attempt is rejected with an error.

DeleteProfile RPC structure

```
rpc DeleteProfile(DeleteProfileRequest) returns (DeleteProfileResponse);
```

9.2.5 GetProfileList RPC

The Certz GetProfileList RPC allows a client to retrieve a list of the TLS profile IDs that are present on a target node.

GetProfileList RPC structure

```
rpc GetProfileList(GetProfileListRequest) returns (GetProfileListResponse);
```

9.2.6 gNSI Certz tools commands

To allow the gNSI Certz RPC behavior to be invoked in the CLI or by a gNMI or JSON-RPC client, SR Linux supports the following tools commands:

- **tools system tls server-profile** *<name>* **certz rotate**
- **tools system tls server-profile** *<name>* **certz remove**

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

9.2.6.1 Rotating a Certz profile using the tools command

Procedure

To rotate a profile on the system, use the **tools system tls server-profile** <*name*> **certz rotate** command, which supports the following parameters:

- **certificate**: specifies the new certificate to use
- **created-on**: sets the created on value for the new policy
- **crl**: specifies a bundle of certificates to add to the CRL
- **key**: specifies the new private key to use
- **trust-anchor**: specifies a certificate chain to use as a trust anchor
- **version**: specifies the version string to store with the policy

Example: Rotate Certz profile using tools command

The following example rotates the **test-certz-profile** with certificate **test-cert**, key **test-key**, CRL **test-crl**, trust anchor **test-anchor**, and assigned version of **1**.

```
--{ candidate shared default }--[  ]--
# tools system tls server-profile test-certz-profile certz rotate certificate test-cert
 key test-key crl test-crl trust-anchor test-anchor version 1
/system/tls/server-profile[name=test-certz-profile]:
    Certz SSL profile 'test-certz-profile' has been added

/system/tls/server-profile[name=test-certz-profile]:
    Certz SSL profile 'test-certz-profile' has been rotated and finalized (version '1'
 created on '2024-02-16T16:20:03.000Z')
```

9.2.6.2 Removing a Certz profile using the tools command

Procedure

To remove a profile from the system, use the **tools system tls server-profile certz remove** command.

Example: Remove a Certz SSL profile using tools command

```
--{ candidate shared default }--[  ]--
# tools system tls server-profile test-certz-profile certz remove
/system/tls/server-profile[name=test-certz-profile]:
    Certz SSL profile 'test-certz-profile' has been removed
```

9.2.7 gNSI Certz state commands

Certz state paths are available under **system tls server-profile**. These paths indicate the certificate, trust bundle, and CRL in use by each specific instance of a gRPC server.

To display the currently installed Certz policy artifacts, use the following commands:

- **info from state system tls server-profile** <*name*> **certz certificate**
- **info from state system tls server-profile** <*name*> **certz crl**

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- **info from state system tls server-profile** *<name>* **certz ssl-profile-id**
- **info from state system tls server-profile** *<name>* **certz trust-anchor**

Example: Info from state for empty TLS server profile dyn

```
--{ running }--[  ]--
# info with-context from state system tls server-profile dyn
    system {
        tls {
            server-profile dyn {
                dynamic true
                certz {
                    ssl-profile-id dyn
                }
            }
        }
    }
```

Example: Info from state for rotated TLS server profile dyn

```
--{ running }--[  ]--
# info with-context from state system tls server-profile dyn
    system {
        tls {
            server-profile dyn {
                key $aes1$ATQxUhdz4QGeu28=$HrE...
                certificate "-----BEGIN CERTIFICATE----- MIIGW...  ----END CERTIFICATE----
-"
                authenticate-client true
                trust-anchor "-----BEGIN CERTIFICATE----- MIIFe...  -----END CERTIFICATE--
---"
                dynamic true
                cipher-list [
                    ecdhe-ecdsa-aes256-gcm-sha384
                    ecdhe-ecdsa-aes128-gcm-sha256
                    ecdhe-rsa-aes256-gcm-sha384
                    ecdhe-rsa-aes128-gcm-sha256
                ]
                certz {
                    ssl-profile-id dyn
                    certificate {
                        version 7
                        created-on "2023-08-19T08:56:45.000Z (6 months ago)"
                    }
                    trust-anchor {
                        version 9
                        created-on "2023-08-19T14:53:09.000Z (6 months ago)"
                    }
                }
            }
        }
    }
```

9.3 gNSI Acctz service

The gNSI Acctz service allows clients to stream accounting records from a target node. This service acts as an auditing mechanism of what changes were attempted or completed on that node.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Supported RPCs

SR Linux supports the following Acctz service RPC:

- RecordSubscribe RPC

9.3.1 RecordSubscribe RPC

The RecordSubscribe RPC is used in the Acctz service to accept a stream of RecordRequest messages and then return a stream of RecordResponse messages.

The RecordSubscribe RPC also maintains a history of accounting records that can be retrieved periodically by connected collector devices.

A SessionInfo message is returned with every RecordResponse message. The SessionInfo message contains information about the context in which the service record was generated.

The SessionInfo message contains the following information:

- `local_address` – local end IPv4 or IPv6 address of the target node that the session is connected to
- `local_port` – local end TCP or UDP port on the target node that the session is connected to
- `remote_address` – remote end IPv4 or IPv6 address that the session is connected from
- `remote_port` – remote end TCP or UDP port that the session is connected from
- `ip_proto` – IP protocol number used by the session
- `channel_id` – used when multiple channels are multiplexed over a single connection

RecordSubscribe RPC structure

```
rpc RecordSubscribe(stream RecordRequest) returns (stream RecordResponse);
```

9.4 gNSI EnrollZ service

Note: gNSI EnrollZ is supported on 7250 IXR-6e and 7250 IXR-10e CPM4 with Root of Trust, 7250 IXR-X1b/X3b, 7250 IXR-X4, and 7250 IXR-18e systems.

The gNSI EnrollZ service allows clients to:

- retrieve the Initial Device Identity (IDevID) and Initial Attestation Key (IAK) certificates from each router control card
- store the owner Initial Device Identity (oIDevID) and owner Initial Attestation Key (oIAK) in each router control card

Note: The oIDevID and oIAK are certificates signed by the owner Certificate Authority (CA) for the associated IDevID and IAK public keys, provisioned by Nokia in the router control card TPM (Trusted Platform Module).

Supported RPCs

SR Linux supports the following EnrollZ service RPCs:

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- GetIAKCert RPC
- RotateOIakCert RPC

9.4.1 GetIAKCert RPC

The GetIAKCert RPC is used in the EnrollZ service to retrieve the Initial Device Identity (IDevID) and Initial Attestation Key (IAK) certificates from each router control card.

The GetIAKCert RPC uses the structure GetIakCertRequest for the request data, which includes the selection of a control card.

The GetIAKCert RPC uses the structure GetIakCertResponse for the response data, which contains the following information:

- **control_card_id**: the control card from which the information originated
- **idevid_cert**: the IDevID certificate
- **iak_cert**: the IAK certificate

GetIAKCert RPC structure

```
rpc GetIakCert(GetIakCertRequest) returns (GetIakCertResponse);
```

9.4.2 RotateOIakCert RPC

The RotateOIakCert RPC stores the owner Initial Device Identity (oIDevID) and owner Initial Attestation Key (oIAK) in each router control card. The oIDevID and oIAK are certificates signed by the owner CA for the associated IDevID and IAK public keys in the same control card.

The RotateOIakCert RPC uses the structure RotateOIakCertRequest for the request data, which includes the following information:

- **control_card_selection**: the selection of the control card
- **oidevid_cert**: the owner IDevID certificate
- **oiak_cert**: the owner IAK certificate
- **ssl_profile_id**: TLS server-profile name

When the **ssl_profile_id** is specified in RotateOIakCertRequest, the TLS server-profile configuration is changed to `use-tpm-devid oidevid`. For configuration details, see the section "Configuring a TLS profile with TPM Device Identity" in SR Linux Configuration Basics Guide.

The RotateOIakCert RPC uses the structure RotateOIakCertResponse for the response data.

RotateOIakCert RPC structure

```
rpc RotateOIakCert(RotateOIakCertRequest) returns (RotateOIakCertResponse);
```

Certificate chain support for oIAK and oIDevID

SR Linux supports a certificate chain for the oIAK and oIDevID certificates returned in EnrollZ RPC.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

The **oiak_cert** and **oidevid_cert** are PEM-encoded strings supporting a single end-identity certificate or a chain of up to six X.509 certificates — one end-identity certificate and up to five parent certificates.

The certificate chain is ordered and starts with the end-identity certificate. Each following certificate must directly certify the one preceding it.

9.5 gNSI AttestZ service

Note: gNSI AttestZ is supported on 7250 IXR-6e and 7250 IXR-10e CPM4 with Root of Trust, 7250 IXR-X1b/X3b, 7250 IXR-X4, and 7250 IXR-18e systems.

The gNSI AttestZ service allows clients to verify the router boot integrity via remote attestation. This capability relies on Measured Boot and requires control cards with TPM provisioned with an Initial Attestation Key (IAK) to sign the measurements. For information on Measured Boot, see the section "Measured Boot" in *SR Linux Software Installation Guide*.

The client verifier can request each control card to perform a TPM quote operation by selecting the Platform Configuration Registers (PCRs) of interest and a random nonce value to verify the boot integrity. The result quote and its signature are generated by the TPM of the control card. The signature is created using the TPM IAK private key so the verifier can ensure that the TPM quote originates from a trusted Nokia control card TPM by verifying the signature chain against the Nokia Factory CA.

Supported RPC

SR Linux supports the following AttestZ service RPC: AttestRequestRPC.

9.5.1 AttestRequest RPC

The AttestRequest RPC uses the structure AttestRequest for the request data, which includes the following information:

- **control_card_id**: the control card the info originated from
- **nonce**: a random nonce value used in the PCR quote to prevent replay attacks
- **hash_algo**: a hash algorithm for the PCR digest
- **pcr_indices**: the PCR selection

The AttestRequest RPC uses the structure AttestResponse for the response data, which contains the following information:

- **control_card_selection**: the selection of the control card
- **tpms_quote_info**: the TPM quote message, which includes the hash of the selected PCRs and additional TPM information
- **quote_signature**: the TPM signature over the quote message
- **oiak_cert**: the oIAK certificate
- **oidevid_cert**: the oDevID certificate

AttestRequest RPC structure

```
rpc Attest(AttestRequest) returns (AttestResponse);
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

9.6 gNSI Pathz service

Note: gNSI Pathz is supported on 7220 IXR-D*x*/Hx, 7250 IXR-6/10/6e/10e/X1b/X3b and 7215 IXS-A1 platforms.

The gNSI Pathz service allows a client to configure and manage an authorization policy defining which users are permitted access to specific gNMI paths. Only one OpenConfig gNMI path-based authorization policy can be active per target node.

The gNSI Pathz service implements a best match approach to matching paths in a request.

The Pathz policy can specify a gNMI origin such as OpenConfig or native. It can also support the use of mixed origin in a single policy. If no rule exists for an origin, Pathz assumes an implicit deny.

Typical Pathz service steps

In the typical Pathz service process, the client performs the following steps using the Rotate, Probe, and Get RPCs:

1. Upload the authorization policy by sending an UploadRequest to the target node using the Rotate RPC.

   Note: The policy specified in the UploadRequest is not activated until a FinalizeRequest is received.

2. Verify that the authorized users can access the required paths and that non-authorized users are denied by sending a ProbeRequest using the Probe RPC.
3. Lock in the updates by sending a FinalizeRequest using the Rotate RPC. The new policy is made active when the FinalizeRequest is received.
4. As required, retrieve details of the currently active policy by sending a GetRequest using the Get RPC.

Note: Unlike other gNSI services, the Probe and Get RPCs allow the client to indicate the specific policy instance that the request is targeting. The following PolicyInstance values are supported:

- **POLICY_INSTANCE_ACTIVE** – indicates the active policy
- **POLICY_INSTANCE_SANDBOX** – indicates the pending policy

9.6.1 Authorization policy

The Pathz gNMI authorization policy provides a framework for controlling which gNMI paths a user can access.

The authorization policy is deployed to a network device and has the ability to define the following:

- policy rule – defines a single authorization policy
- groups of users – logically groups users in the administrative domain
- users – references individuals in rules or group definitions

Match rules allow a match against the following:

- a user or a group, but not both
- a gNMI path

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- the access mode (read or write)

When evaluating a user's authorization against the policy rules, Pathz selects the best, most specific matching path using the following rules, in order:

1. Prefer longer matches over shorter matches.
2. Prefer definite keys over wildcard keys. A rule containing more definite keys is selected over a rule with fewer definite keys.
3. Prefer users over groups. A rule matching a user is selected over a rule matching a group that the user belongs to.
4. Prefer deny actions over permit actions. If all preceding rules are equal, the rule with the deny action is preferred.

If a matching rule does not exist in the policy, an implicit deny is assumed.

Example: Authorization policy matching example

The following shows two rules with similar paths which differ only with respect to attribute wildcarding.

```
/a/b[key=FOO]/c/d
/a/b[key=*]/c/d
```

Example: Authorization protobuf example

The following shows an example where user Stevie is permitted access to the defined path based on the policy rule.

```
version: "UUID-1234-123123-123123"
created_on: 1234567890
# Define 2 groups.
group {
  name: "family-group"
  members { name: "stevie" }
  members { name: "brian" }
}
group {
  name: "test-group"
  members { name: "crusty" }
  members { name: "the-clown" }
}
# Action stevie to access /this/is/a/message_path in a READ manner.
policy {
  id: "one"
  path {
    origin: "foo"
    elem { name: "this" }
    elem { name: "is" }
    elem { name: "a" }
    elem { name: "message_path" }
  }
  action: PERMIT
  mode: READ
  user { name: "stevie" }
}
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

9.6.2 Interactions with role-based access control

The gNSI Pathz service overlaps with existing mgmt_server role-based access control (RBAC) behavior. The following conditions allow Pathz and RBAC to coexist:

- No Pathz policy exists by default. However, a policy can be injected during Zero Touch Provisioning (ZTP).
- If no Pathz policy is applied, existing roles continue their default behavior.
- If a valid Pathz policy is applied, future gNMI RPCs evaluate against the new policy.

The gNSI Pathz service differs from RBAC in the following ways:

- Pathz can apply policy rules to both groups and users independently.
- Pathz models its own group constructs.
- If no rule in the policy exists, Pathz assumes an implicit deny.
- If two rules match a request equally, the request is rejected instead of allowing the more permissive match.

There are two ways to implement policy rules with Pathz:

- Create one internal group per user with each group name matching the user name, and apply the policy rule to the internal group.
- Assign the native RBAC model directly to the user, where the user is directly assigned a mgmt_server role independent of the aaa_mgr groups.

9.6.3 Supported RPCs

SR Linux supports the following Pathz service RPCs:

- Rotate RPC
- Probe RPC
- Get RPC

9.6.4 Rotate RPC

The Pathz Rotate RPC replaces an existing Pathz authorization policy (or loads an initial policy) on the target node. If the stream is broken or any steps in the process fail, the target node rolls back any changes made by the RPC.

Only one Rotate RPC can be in progress at a time. If a client attempts to call the RPC while another call is already in progress, the second call is rejected with a gRPC error of `UNAVAILABLE`.

During a Rotate RPC, the policy provided in the UploadRequest does not become an active policy until a FinalizeRequest message is received. If the target node responds to the FinalizeRequest message with an error, the Pathz policy updates are rolled back.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Rotate RPC structure

```
rpc Rotate(stream RotateRequest) returns (stream RotateResponse);
```

9.6.5 Probe RPC

During a Rotate operation, the Probe RPC allows a client to test the pending policy before finalizing the policy rotation. The RPC allows the Pathz policy engine on the target node to provide a response to a specified gNMI operation performed on a single gNMI path by a user. The target node returns the response without actually performing the gNMI operation. The RPC can be used to test the active policy at any time.

Probe RPC structure

```
rpc Probe(ProbeRequest) returns (ProbeResponse);
```

9.6.6 Get RPC

The Get RPC returns the gNMI authorization policy requested in the RPC, as well as the policy version and created-on timestamp.

Get RPC structure

```
 rpc Get(GetRequest) returns (GetResponse);
```

9.7 gNSI configuration

SR Linux supports gNSI services using the gRPC server configuration. To enable gNSI support, enable the gRPC server and specify gNSI as one of the enabled gRPC services for a specified network instance (enabled by default on the mgmt network instance).

Like gNMI, the session between the gNSI client and SR Linux can be encrypted using TLS.

gNMI and gNSI servers enabled by default

By default, the gNMI and gNSI servers are enabled in the mgmt network instance.

```
# info with-context system grpc-server mgmt
    system {
        grpc-server mgmt {
            admin-state enable
            default-tls-profile true
            network-instance mgmt
            services [
                gnmi
                gnsi
            ]
        }
    }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

9.7.1 Configuring gNSI services

About this task

As part of the gRPC server configuration, you can also specify which individual gNSI services to enable.

Procedure

To enable gNSI services, use the **system grpc-server** <*name*> **services** command.

Note: If you enter **services [gnsi]** all gNSI services are enabled. To enable an individual service only, enter for example **services [gnsi.authz]** or **services [gnsi.certz]**, and omit the **[gnsi]** parameter.

Example: Enable gNSI services

```
# info with-context system grpc-server mgmt
    system {
        grpc-server mgmt {
            admin-state enable
            network-instance mgmt
            services [
                gnsi
            ]
        }
    }
```

See the "Management servers" chapter in the SR Linux Configuration Basics Guide for related information about how to configure the gRPC server.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

10 JSON interface

SR Linux provides a JSON-based Remote Procedure Call (JSON-RPC) interface for CLI commands, configuration, and YANG-modeled actions. It enables you to retrieve and modify configuration and state, execute YANG-modeled actions, and receive responses in JSON format.

If output from a command cannot be displayed in JSON, the text output is wrapped in JSON to allow the application calling the API to retrieve the output. During configuration, if a TCP port is in use when the JSON-RPC server attempts to bind to it, the commit fails. The JSON-RPC supports both normal paths, as well as XPATHs.

10.1 JSON message structure

JSON RPC requires a specific message structure. The jsonrpc version, ID, method, and params are required in all requests. For example:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "get",
  "params": {
  }
}
```

Within these required elements can be additional mandatory and conditional elements, as described in the following table.

Table 23: Required JSON request structure

Required request elements	Description
`jsonrpc`	Version, which must be "2.0". No other JSON RPC versions are currently supported.
`id`	Client-provided integer. The JSON RPC responds with the same ID, which allows the client to match requests to responses when there are concurrent requests.
`method`	Defines the method accessed with the JSON RPC. Supported options are as follows. • `get` • `set` • `validate` • `cli`

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Required request elements	Description
	• `action`
`params`	Defines a container for any parameters related to the request. The type of parameter is dependent on the method used.

Related topics

method options

params options

10.1.1 method options

The following table defines supported JSON RPC method options.

Table 24: JSON RPC method options

Method option	Description
`get`	Used to retrieve configuration and state details from the system. The `get` method can be used with candidate, running, and state datastores, but cannot be used with the tools datastore.
`set`	Used to set configuration in the candidate datastore. For a YANG-modeled action, use the `action` method,
`validate`	Used to verify that the system accepts a configuration transaction before applying it to the system.
`cli`	Used to execute CLI commands. The `get` and `set` methods are restricted to accessing data structures via the YANG models, but the `cli` method can access any commands added to the system via python plug-ins or aliases. Both the `cli` and `action` methods are valid currently. However, automations that use `cli` method to execute operational commands can be migrated to the `action` method over time to provide structured data input and output aligned with the YANG models.
action	Used to execute a YANG modeled action with input and output encoding aligned with the YANG schema. The path identifies the YANG `action` node and includes an exact value for every list key. An optional JSON payload specifies the action `input` leaves. The response contains the modeled

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Method option	Description
	action `output` and does not include the `srl-unstructured-output` field. See Unstructured output for more information. If the action fails because of authorization, validation, or handler errors, the server returns a JSON-RPC error. Each request executes one action.

10.1.2 params options

The following table defines valid JSON RPC params options.

Table 25: JSON RPC params options

params option	Descriptions
action	**action** command - Conditional mandatory; used with the set and validate methods. Supported options are: • **replace** — Replaces the entire configuration within a specific context with the supplied configuration; equivalent to a delete/update. • **update** — Updates a leaf or container with the specified value. • **delete** — Deletes a leaf or container. All children beneath the parent are removed from the system. **Note:** When the **action** command is used with the tools datastore, update is the only supported option. **Notice:** The JSON-RPC `action` method and the **action** command method serve different purposes. The `action` params command specifies the operation to perform in the JSON-RPC such as replacing, updating, or deleting configuration data.
path	**path** command - Mandatory with the get, set, validate, and action methods. This value is a string that follows the gNMI path specification[4] in human-readable format: • "/" separates nodes • keys are specified using [<key-name>=<key-value>] • if key-value contains "]", it must be escaped ("\" before the "]" • fields end with "." or ",value>" Example: /interface[name=mgmt0]

[4] gNMI path specification reference: https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-path-conventions.md

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

params option	Descriptions
path-keywords	**path-keywords** command - Optional; used to substitute named parameters with the path field. More than one keyword can be used with each path.
payload	**payload** command - Optional. Used with the `action` method. A JSON object containing the YANG structured instance data from the action `input` statement.
datastore	**datastore** command - Optional; selects the datastore to perform the method against. Supported options are: • **candidate** — Used to change the configuration of the system with the get, set, and validate methods; default datastore is used if the datastore parameter is not provided. • **running** — Used to retrieve the active configuration with the get method. • **state** — Used to retrieve the running (active) configuration along with the operational state. • **tools** — Used to perform operational tasks on the system; only supported with the update action command and the set method. For a YANG modeled action, use the `action` or `cli` method.
recursive	**recursive** command - Optional; a Boolean used to retrieve children underneath the specific path. The default = true.
include-field-defaults	**include-field-defaults** command - Optional; a Boolean used to show all fields, regardless if they have a directory configured or are operating at their default setting. The default = false.
output-format - Optional. Defines the output format as: • json • text • table	**output-format** command - cli method — **output-format** (optional, default: json): Specifies the output format for ordinary CLI commands. Supported values are json, text, table, yaml, xml, and flat. This parameter does not affect a tools or action line. cli method — **tools-output-format** (optional): Specifies the output format for a tools command. Supported values are json and unstructured-output. action method — **output-format** (optional, default: json): Specifies the output format of the action result. The action method does not support a separate **tools-output-format** parameter.

10.2 JSON responses

The JSON RPC returns one entry for each command that was executed. For methods that do not contain response (other than acknowledging the command), a response is returned, but with an empty list.

When the `cli` method is used, each executed command returns an individual response.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Compressed JSON responses using gzip

If a client request specifies to use HTTP compression, the JSON RPC server compresses the responses using gzip.

10.3 Candidate mode

JSON uses its own private exclusive candidate that restricts other users or services from making simultaneous changes to a configuration. If another exclusive session is already active, any commits to the configuration using the JSON-RPC set method fail with an error.

The JSON-RPC server uses the private exclusive candidate name **jsonrpc-**<*n*>, where <*n*> is a 32-bit number starting at 1 that increments for every request received, but resets on a JSON-RPC server restart.

10.4 Logical expressions

For logical expressions, support is provided for the asterisk ("*") which can be used to reference all keys within a list. This is not applicable for **action** method.

10.5 Interactive CLI commands

A small subset of command require interactive user input. These commands cannot be executed using any interface except the CLI.

10.6 JSON examples

The following sections provides JSON examples (both requests and responses) for `get`, `set`, `delete`, `validate`, `cli` and `action` method options.

10.6.1 JSON get examples

The get method allows you to retrieve configuration and state details from the system. The following examples are shown:

- get method using the path, datastore, and recursive commands with the recursive option set to true (to retrieve children underneath the specific path)
- multiple get request where multiple commands are executed with a single request

Example: Single get with recursive request

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "get",
  "params": {
    "commands": [
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
      {
        "path": "/interface[name=mgmt0]",
        "datastore": "state",
        "recursive": true
      }
    ]
  }
}
```

Response (single get with recursive)

```
{
  "result": [
    {
      "name": "mgmt0",
      "admin-state": "enable",
      "mtu": 1514,
      "ifindex": 524304383,
      "oper-state": "up",
      "last-change": "2019-07-12T16:53:39.291Z",
      "statistics": {
        "in-octets": "4545395",
        "in-unicast-pkts": "1178",
        "in-broadcast-pkts": "130",
        "in-multicast-pkts": "27560",
        "in-discards": "0",
        "in-errors": "0",
        "in-unknown-protos": "0",
        "in-fcs-errors": "0",
        "out-octets": "1735990",
        "out-unicast-pkts": "1125",
        "out-broadcast-pkts": "38",
        "out-multicast-pkts": "9187",
        "out-discards": "0",
        "out-errors": "0",
        "carrier-transitions": "1"
      },
      "ethernet": {
        "hw-mac-address": "02:42:AC:12:00:05",
        "statistics": {
          "in-mac-control-frames": "0",
          "in-mac-pause-frames": "0",
          "in-oversize-frames": "0",
          "in-jabber-frames": "0",
          "in-fragment-frames": "0",
          "in-crc-errors": "0",
          "out-mac-control-frames": "0",
          "out-mac-pause-frames": "0"
        }
      },
      "subinterface": [
        {
          "index": 0,
          "admin-state": "enable",
          "ip-mtu": 1500,
          "ifindex": 524288000,
          "oper-state": "up",
          "last-change": "2019-07-12T16:53:39.291Z",
          "ipv4": {
            "allow-directed-broadcast": false,
            "dhcp-client": true,
            "address": [
              {
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
            "ip-prefix": "172.18.0.5/24",
            "origin": "dhcp"
          }
        ],
        "srl_nokia-interfaces-nbr:arp": {
          "timeout": 14400,
          "neighbor": [
            {
              "ipv4-address": "172.18.0.1",
              "link-layer-address": "02:42:90:06:D7:26",
              "origin": "dynamic",
              "expiration-time": "2019-07-15T21:37:28.987Z"
            },
            {
              "ipv4-address": "172.18.0.2",
              "link-layer-address": "02:42:AC:12:00:02",
              "origin": "dynamic",
              "expiration-time": "2019-07-16T00:44:17.673Z"
            },
            {
              "ipv4-address": "172.18.0.3",
              "link-layer-address": "02:42:AC:12:00:03",
              "origin": "dynamic",
              "expiration-time": "2019-07-16T01:20:48.600Z"
            },
            {
              "ipv4-address": "172.18.0.4",
              "link-layer-address": "02:42:AC:12:00:04",
              "origin": "dynamic",
              "expiration-time": "2019-07-16T01:20:48.597Z"
            },
            {
              "ipv4-address": "172.18.0.6",
              "link-layer-address": "02:42:AC:12:00:06",
              "origin": "dynamic",
              "expiration-time": "2019-07-16T01:20:48.599Z"
            }
          ]
        }
      },
      "ipv6": {
        "dhcp-client": true,
        "address": [
          {
            "ip-prefix": "2001:172:18::5/80",
            "origin": "dhcp",
            "status": "preferred"
          },
          {
            "ip-prefix": "fe80::42:acff:fe12:5/64",
            "origin": "link-layer",
            "status": "preferred"
          }
        ],
        "srl_nokia-interfaces-nbr:neighbor-discovery": {
          "dup-addr-detect": true,
          "reachable-time": 30,
          "stale-time": 14400
        }
      }
    }
  ]
}
],
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
  "id": 0,
  "jsonrpc": "2.0"
}
```

Example: Multiple get request

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "get",
  "params": {
    "commands": [
      {
        "path": "/interface[name=mgmt0]",
        "datastore": "state",
        "recursive": false
      },
      {
        "path": "/interface[name=ethernet-1/10]",
        "datastore": "state",
        "recursive": false
      }
    ]
  }
}
```

Response (multiple get)

```
{
  "result": [
    {
      "name": "mgmt0",
      "admin-state": "enable",
      "mtu": 1514,
      "ifindex": 524304383,
      "oper-state": "up",
      "last-change": "2019-07-12T16:53:39.291Z"
    },
    {
      "name": "ethernet-1/10",
      "description": "dut2-dut4-2",
      "admin-state": "enable",
      "mtu": 9232,
      "ifindex": 180223,
      "oper-state": "up",
      "last-change": "2019-07-12T16:54:15.602Z"
    }
  ],
  "id": 0,
  "jsonrpc": "2.0"
}
```

10.6.2 JSON set examples

The set method allows you to set candidate configuration. The following examples are shown:

- multiple set method using update and replace
- set method using **path-keywords**

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- set method using an alternative update

Example: Multiple set with update and replace request

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "set",
  "params": {
     "commands": [
        {
           "action": "update",
           "path": "/interface[name=mgmt0]/description:my-description"
        },
        {
           "action": "replace",
           "path": "/interface[name=mgmt0]/subinterface[index=0]/description:my-
subdescription"
        }
     ]
  }
}
```

Response (multiple set)

```
{
  "result": {},
  "id": 0,
  "jsonrpc": "2.0"
}
```

Example: set using path keywords request

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "set",
  "params": {
    "commands": [
      {
        "action": "update",
        "path": "/interface[name={name}]/description:my-description",
        "path-keywords": {
          "name": "mgmt0"
        }
      },
      {
        "action": "replace",
        "path": "/interface[name={name}]/subinterface[index={index}]/description:my-
subdescription",
        "path-keywords": {
          "name": "mgmt0",
          "index": "0"
        }
      }
    ]
  }
}
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Response (set using path-keywords)

```
{
  "result": {},
  "id": 0,
  "jsonrpc": "2.0"
}
```

Example: set using alternative update (specifying a value) request

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "set",
  "params": {
    "commands": [
      {
        "action": "update",
        "path": "/interface[name=mgmt0]",
        "value": {
          "description": "my-description",
          "subinterface": {
            "index": "0",
            "description": "my-subdescription"
          }
        }
      }
    ]
  }
}
```

Response (set using alternative update)

```
{
  "result": {},
  "id": 0,
  "jsonrpc": "2.0"
}
```

10.6.3 JSON delete example

The set method allows you to use an action delete command to delete nodes or leafs within a configuration.

The following example shows a multiple set delete request to delete the specified paths.

Example: Multiple set method delete request

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "set",
  "params": {
    "commands": [
      {
        "action": "delete",
        "path": "/interface[name=mgmt0]/description"
      },
      {
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
        "action": "delete",
        "path": "/interface[name=mgmt0]/subinterface[index=0]/description"
      }
    ]
  }
}
```

Response (multiple set delete)

```
{
  "result": {},
  "id": 0,
  "jsonrpc": "2.0"
}
```

10.6.4 JSON validate example

The validate method allows you to verify that the system will accept a configuration transaction before applying it to the system. The ‘delete’, ‘replace’, and ‘update’ actions can be used with the validate method. The following examples are shown:

- validate request to delete a specified path.
- validate request to update and replace a specified path.

Example: validate a delete request

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "validate",
  "params": {
    "commands": [
      {
        "action": "delete",
        "path": "/interface[name=mgmt0]/description"
      }
    ],
    "datastore": "candidate"
  }
}
```

Response (validate delete)

```
{
  "result": {},
  "id": 0,
  "jsonrpc": "2.0"
}
```

Example: validate an update and replace request

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "validate",
    "params": {
        "commands": [
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
            {
                "action": "update",
                "path": "/interface[name=mgmt0]/description:my-description"
            },
            {
                "action": "replace",
                "path": "/interface[name=mgmt0]/subinterface[index=0]
 /description:my-subdescription"
            }
        ]
    }
}
```

Response (validate update and delete)

```
{
  "result": {},
  "id": 0,
  "jsonrpc": "2.0"
}
```

10.6.5 JSON CLI example

The `cli` method allows you to run CLI commands, including the global tools command for a YANG modeled action. The tools command uses the YANG path of the action, in the same form as the MD-CLI. The `cli` method can also run commands added to the system with Python plug-ins or aliases.

The following examples are shown:

- JSON `cli` command request.
- JSON `cli` command requesting the output format as text.

You can optionally define these output formats: json, text, or table. JSON is the default.

Example: CLI method input request

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cli",
  "params": {
    "commands": [
      "enter candidate",
      "info interface mgmt0"
    ]
  }
}
```

Response (CLI input)

```
{
  "result": [
    {},
    {
      "interface": [
        {
          "name": "mgmt0",
          "description": "my-description",
          "admin-state": "enable",
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
          "subinterface": [
            {
              "index": 0,
              "description": "my-subdescription",
              "admin-state": "enable",
              "ipv4": {
                "dhcp-client": true
              },
              "ipv6": {
                "dhcp-client": true
              }
            }
          ]
        }
      ]
    }
  ],
  "id": 0,
  "jsonrpc": "2.0"
}
```

Example: CLI method input request with output formatting

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cli",
  "params": {
    "commands": [
      "enter candidate",
      "info interface mgmt0"
    ],
    "output-format": "text"
  }
}
```

Response (CLI with output formatting)

```
{
  "result": [
    "",
    "    interface mgmt0 {\n      description my-description\n
admin-state enable\n              subinterface 0 {\n
description my-subdescription\n   admin-state enable\n       ipv4 {\n
dhcp-client true\n                }\n                        ipv6 {\n
dhcp-client true\n                }\n                        }\n      }\n"
  ],
  "id": 0,
  "jsonrpc": "2.0"
}
```

10.6.6 JSON action examples

The action method executes a YANG-modeled action using structured JSON input and output that conform to the YANG schema. The encoding is not CLI text. This method is separate from the `action` field on the `set` and `validate` methods.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Set `path` to the action node. Use an absolute path or an origin-qualified path. Include the exact key of every keyed list in the path.

Optionally set `input` to a JSON object mapped from the YANG input leaves. The `output` is a JSON object mapped from the YANG output leaves. The response does not include `srl-unstructured-output`. See *SR Linux CLI Plug-In Guide* for more information about `srl-unstructured-output`.

Errors are returned as a JSON-RPC error object with the same failure semantics as other management APIs, including authorization, validation, and handler errors.

Example: Action method request - IPv4

```
ipv4:
Request:
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "action",
  "params": {
    "commands": [
      {
        "path": "/system/ipfix/flow-cache/dump-flows",
        "payload": {
          "ipv4": [null]
        }
      }
    ],
    "output-format": "json"
  }
}
```

Response (action method)

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": [
    {
      "flow-dump": "type: 1\nNo IPv4 unicast flow data has been recorded by cflowd\n"
    }
  ]
}
```

Example: Action method request - IPv6

```
ipv6:

Request:
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "action",
  "params": {
    "commands": [
      {
        "path": "/system/ipfix/flow-cache/dump-flows",
        "payload": {
          "ipv6": [null]
        }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
      }
    ],
    "output-format": "json"
  }
}
```

Response

```
Response:
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": [
    {
      "flow-dump": "type: 2\n\nThe top 20 IPv6 unicast flows seen by cflowd are:\n\
n     Current Time: 10/13/2026 23:35:00\nLast Cleared Time: 10/13/2026 21:52:17\n
 ifIndexContext: global\n\nSrcIP                                          Intf/Ingr    S-
Port   I-vRtrID    ToS\nDstIP                                      Intf/Egr     D-Port
   E-vRtrID  Flags\n Nexthop                                 Pkts         Pkt-Size
 Proto      Time\n-------------------------------------------------------------------
-------------\n3ffe::13:13:13:3                          98328        0        10284
      0x00\n3ffe::12:12:12:2                        0            0        0
     0x00\n ::                                    3418         238      58
     34\n3ffe::13:13:13:3                        98328        0        10284
  0x00\n3ffe::12:12:12:2                       0            0        0
 0x00\n ::                                    3400         238      58
 33\n3ffe::13:13:13:3                       98328        0        10284      0x00\
n3ffe::12:12:12:2                    0            0        0          0x00\n ::
                                  3001         238      58           30\n---------------
-----------------------------------------------------------------\n\n"
    }
  ]
}
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

11 NETCONF

This section describes the use of the Network Configuration Protocol (NETCONF) in SR Linux.

Related topics

Using the diff command

11.1 NETCONF overview

NETCONF is a standardized IETF configuration management protocol specified in RFC 6241. It is secure, connection-oriented, and runs on top of the SSHv2 transport protocol as specified in RFC 6242. NETCONF is an XML-based protocol that can be used as an alternative to the CLI or gNMI for managing SR Linux.

NETCONF uses Remote Procedure Call (RPC) messaging to facilitate communication between a NETCONF client and the NETCONF server that is running on SR Linux. The RPC operation, and the associated 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 message interactions.

The NETCONF interface on SR Linux supports configuration, operational state, and YANG-modeled operations (`action` statements). NETCONF notifications are not supported. See YANG-modeled operations - NETCONF.

11.1.1 Transport and sessions

NETCONF uses the SSHv2 secure transport as defined in RFC 6242.

The NETCONF protocol standard specifies a default port of 830 over TCP as registered in IANA (see *Service Name and Transport Protocol Port Number Registry*). The port must be specifically configured in the SSH server configuration for NETCONF. The default port is permitted inbound in the default CPM filter.

NETCONF sessions do not time out automatically and are not subject to the CLI and SSH session timeouts. Operators can disconnect sessions manually as needed.

A locally configured user (including `admin` and `linuxadmin`) or a remote user assigned the `netconf` role can authenticate with the system to establish a NETCONF session. For more information about how to establish a session, see Initiating a connection to a NETCONF server.

The SR Linux NETCONF server supports:

- binding to one or more network instances, including support for:
- use of SSH public keys and certificates
- multiple SSH channels (additional NETCONF sessions created as required over additional SSH channels)

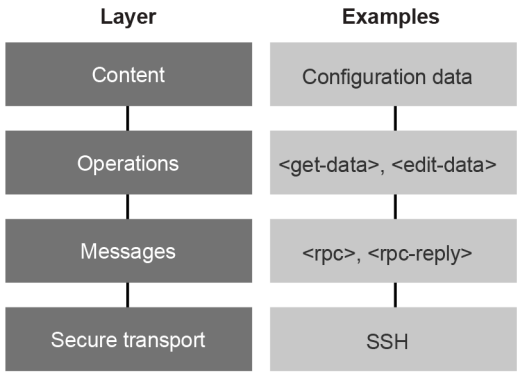
© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

11.1.2 NETCONF in SR Linux

In SR Linux, the NETCONF server is implemented as the `netconf_mgr` process; it owns the `nokia_srl-netconf-server.yang`, `srl_nokia-tools-netconf-server.yang`, `ietf-netconf-monitoring.yang`, and `ietf-yang-library.yang` YANG modules. The `netconf-mgr` process operates as the `netconfmgr` user. Functionally, the NETCONF server becomes a client of `mgmt-server` and in the `netconf-server` YANG module, there is s pointer to the `ssh-server` context to configure the SSH transport.

The NETCONF protocol operates in the following distinct layers:

Figure 9: NETCONF layers



sw4416

Overview of the SR Linux NETCONF implementation

The SR Linux NETCONF implementation covers configuration and state management and has the following behavior:

- supports SSHv2 transport as specified in RFC 6242
- supports only XML encoding

  Every communication with the NETCONF server takes the form of an XML document. An XML document is a correctly formatted and encoded set of YANG validated XML statements. Any invalid XML returns a `malformed-message` error. Every XML document can optionally contain an XML specification, for example: `<?xml version="1.0"?>`.
- does not support notifications over NETCONF
- does not support partial locks as defined in RFC 5717
- does not support XPath filtering

The NETCONF server implements the `ietf-netconf.yang` model as defined in RFC 6241.

11.1.3 YANG-modeled operations - NETCONF

SR Linux YANG modules support defining action statements (YANG 1.1, RFC 7950), in addition to managing configuration and operational state. See YANG-modeled operations for more information. An

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

action runs a one-time operation on the device. When the YANG model defines them, the action accepts optional input parameters and can optionally return structured output in the NETCONF reply.

Over NETCONF, you invoke actions using the standard YANG 1.1 <action> element inside an <rpc> message. SR Linux does not define additional NETCONF RPC operations for YANG-modeled actions.

When an action completes successfully, the NETCONF server returns an <rpc-reply> that contains only the structured output defined in the YANG model for that action. If there is no structured output, NETCONF returns the ok node within <rpc-reply>.

The set of actions available on a system depends on the installed YANG modules. See the *SR Linux Data Model Reference Guide* for action definitions in the data model.

11.1.3.1 Invoking a YANG action

You can invoke an action over NETCONF using the YANG 1.1 <action> element inside an <rpc> message.

A YANG action optionally defines its input parameters within an input container (for example, `action soft-clear { input { leaf direction { ... } } }`). This input container is a logical grouping within the model.

In the actual NETCONF RPC request, the input parameters are placed directly under the action element itself, not nested within an <input> element.

For example, consider a YANG model defining a BGP neighbor **soft-clear** action with a `direction` input:

```
action soft-clear {
  input {
    leaf direction {
      type enumeration {
        enum in;
        enum out;
        enum both;
      }
      default "both";
    }
  }
  // No output container.
}
```

The **soft-clear** action is defined under the BGP neighbor in the YANG data tree. To invoke the action, the NETCONF <action> element must contain the complete path to the target neighbor instance. The path includes the key values required to identify each keyed list instance (for example, the `network-instance` name and the BGP neighbor `peer-address`).

The following example shows the NETCONF RPC request to invoke this **soft-clear** action for BGP neighbor 192.0.2.1 in network-instance **default**. Because the YANG default for `direction` is `both`, the `direction` leaf is omitted and the **soft-clear** element has no child elements:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <network-instance xmlns="urn:nokia.com:srlinux:net-inst">
      <name>default</name>
      <protocols>
        <bgp>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
          <neighbor>
            <peer-address>192.0.2.1</peer-address>
            <soft-clear xmlns="urn:nokia.com:srlinux:bgp:bgp"/>
          </neighbor>
        </bgp>
      </protocols>
    </network-instance>
  </action>
</rpc>
```

Note: The YANG excerpt and XML in these examples are illustrative. Final namespaces and action names must be taken from the YANG modules. The input provided when invoking a YANG action must match the structure and content specified in its corresponding YANG module definition. See the *SR Linux Data Model Reference Guide*.

When an action completes successfully, the NETCONF server returns an `<rpc-reply>` as described in YANG-modeled operations. For details about `<rpc-error>` elements when an action fails, see Error handling.

11.1.3.2 Synchronous operation support

SR Linux executes YANG-modeled actions over NETCONF synchronously, meaning the action completes on the initiating session before the client sends any further NETCONF operations on that same session.

SR Linux does not support asynchronous actions (where the device accepts a request, returns immediately, and exposes results through a separate operations table).

Although the action is synchronous, the server may send action output in the `<rpc-reply>` as it becomes available (see Streamed action output).

11.1.3.3 Streamed action output

Action output may be streamed to the client as it becomes available, either as additional lines of data in the reply or as chunks when chunked framing is in use. For information about chunked framing, see NETCONF server capabilities.

In rare cases, an action may begin returning streamed output and subsequently encounter an unrecoverable error. After action output has begun, it may no longer be possible to return a NETCONF `<rpc-error>` response. In this situation, the NETCONF server terminates the NETCONF session.

11.1.4 NETCONF IETF monitoring

IETF NETCONF monitoring, as defined in RFC 6022 provides information about the current operational status of a NETCONF server. The model contains state information and defines the <get-schema> RPC.

Note: SR Linux does not support the ability to obtain data from IETF-specific configuration models (such as interfaces) in the CLI or gNMI.

```
module: ietf-netconf-monitoring
  +--ro netconf-state
     +--ro capabilities
     |  +--ro capability*   inet:uri
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
    +--ro datastores
    |  +--ro datastore* [name]
    |     +--ro name     netconf-datastore-type
    |     +--ro locks!
    |        +--ro (lock-type)?
    |           +--:(global-lock)
    |           |  +--ro global-lock
    |           |     +--ro locked-by-session    uint32
    |           |     +--ro locked-time          yang:date-and-time
    |           +--:(partial-lock)
    |              +--ro partial-lock* [lock-id]
    |                 +--ro lock-id              uint32
    |                 +--ro locked-by-session    uint32
    |                 +--ro locked-time          yang:date-and-time
    |                 +--ro select*              yang:xpath1.0
    |                 +--ro locked-node*         instance-identifier
    +--ro schemas
    |  +--ro schema* [identifier version format]
    |     +--ro identifier    string
    |     +--ro version       string
    |     +--ro format        identityref
    |     +--ro namespace     inet:uri
    |     +--ro location*     union
    +--ro sessions
    |  +--ro session* [session-id]
    |     +--ro session-id           uint32
    |     +--ro transport            identityref
    |     +--ro username             string
    |     +--ro source-host?         inet:host
    |     +--ro login-time           yang:date-and-time
    |     +--ro in-rpcs?             yang:zero-based-counter32
    |     +--ro in-bad-rpcs?         yang:zero-based-counter32
    |     +--ro out-rpc-errors?      yang:zero-based-counter32
    |     +--ro out-notifications?   yang:zero-based-counter32
    +--ro statistics
       +--ro netconf-start-time?   yang:date-and-time
       +--ro in-bad-hellos?        yang:zero-based-counter32
       +--ro in-sessions?          yang:zero-based-counter32
       +--ro dropped-sessions?     yang:zero-based-counter32
       +--ro in-rpcs?              yang:zero-based-counter32
       +--ro in-bad-rpcs?          yang:zero-based-counter32
       +--ro out-rpc-errors?       yang:zero-based-counter32
       +--ro out-notifications?    yang:zero-based-counter32

rpcs:
  +---x get-schema
     +---w input
     |  +---w identifier    string
     |  +---w version?      string
     |  +---w format?       identityref
     +--ro output
        +--ro data?   <anyxml>
```

The state information is divided into the following containers:

- `capabilities`: lists the capabilities that are currently supported on the server. These capabilities can differ from the capabilities listed in the initial server <`hello`> message as various configuration options can change them during the session, for example, additional YANG modules may have been enabled. Whenever there is a change server capabilities, the NETCONF user should consider restarting the session.
- `datastores`: lists the currently available datastores, for example, candidate and operational.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- `schemas`: lists the schemas currently available; this is similar to the output in the IETF YANG library. For details, see YANG library.
- `sessions`: lists the currently active NETCONF sessions on the NETCONF server.
- `statistics`: provides statistics for the lifecycle of the NETCONF server (instead of the current session).

11.1.5 YANG library

SR Linux supports the YANG library mechanism to identify the available functionality of the NETCONF server including the YANG modules, sub-modules, YANG features and deviations that are implemented or imported by the NETCONF server as well the datastores provided. NETCONF clients can query or cache the YANG library contents and identify whether their cache is out of date using the `content-id` value, which changes each time the set of YANG modules is updated.

SR Linux implements version 1.1 of the IETF YANG library specification, as described in RFC 8525. For backwards compatibility, SR Linux also implements the `modules-state` branch of version 1.0 RFC 7895.

Elements that are not supported in a YANG module (Nokia or third-party) are deviated as not supported and are advertised by the node.

Following is the IETF YANG library version 1.1:

```
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 following sections describe the contents of IETF YANG library version 1.1.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

`module-set` list

The `module-set` list ensures that remote management systems understand and can obtain the correct list of modules that make up the devices schema. This list can change while the system is active based on the configuration and YANG modules available to the system.

The `module-set` list has a name and provides a list of YANG modules that the device has. The YANG modules (whether Nokia or third-party) are grouped as follows:

- `module`: lists implemented modules.
  A module is considered implemented if, after stripping out deviated/not-supported nodes, it meets at least one of the following requirements:
  - has a data node
  - contains an RPC
  - contains an action
  - contains a notification
  - contains a deviation (different from the module has been deviated to remove nodes)
  - contains an identity
- `import-only-module`: lists modules that do not meet any of the requirements listed for an implemented module.

`schema` list

The `schema` list comprises:

- `name` the name of the schema
- `module-set`: points to the `module-set` list

`datastore` list

The `datastore` list comprises:

- `name` the name of the datastore
- `schema`: the schema advertised and used by the specified datastore

`content-id`

The `content-id` value changes whenever there is any change in the YANG modules or module versions that make up the node's entire schema.

The change in content ID means that a change has occurred in the YANG modules that the node is advertising. The change in content ID is a signal to a client to re-obtain a list of YANG modules and then obtain those modules using the <`get-schema`> operation.

SR Linux provides a randomized value for the `content-id` field. There is no expectation made by the standard specification for the `content-id` field to reflect a specific value for a set of modules. The NETCONF session should also be restarted.

If a change in the `content-id` field is detected, it is good practice to restart the NETCONF session.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

11.1.6 Error handling

If an error or warning occurs during a NETCONF operation, the RPC reply message reports the error as <rpc-error>. By specification, the NETCONF server is required to return only one <rpc-error> even if more than one error occurs; however, to assist with troubleshooting the SR Linux NETCONF server reports all errors.

RPC errors are described in RFC 6241. An <rpc-error> message contains the following fields:

Table 26: Fields in the <rpc-error> element

Field	Description
error-app-tag	Where available, identifies whether an error condition is data-model-specific or implementation-specific This element is present when a corresponding error-tag is unavailable for a particular error condition. If the error-app-tag element exists for both data-model-specific and an implementation-specific error conditions, the server uses the data-model-specific value.
error-info	Contains protocol or data-model-specific error content. This element is not present if an appropriate message does not exists for an error condition. The list in *Appendix A* of RFC 6241 defines the mandatory error-info content for each error. After any protocol-mandated content, SR Linux includes application-layer error information in the error-info container. SR Linux includes additional elements to provide extended and implementation-specific debugging information.
error-message	Provides textual information that describes the error. This element is not present if a message does not exist for a particular error condition.
error-path	Provides an absolute XPath expression identifying the element path to the node that is associated with the error being reported.
error-severity	Reports whether the error severity is an error or warning.
error-tag	Identifies the error condition. For more information, see *Appendix A* of RFC 6241. Some operations provide specific error-tag values from this list on an operation-by-operation basis to ensure that the correct error-tag value is selected.
error-type	Identifies in which of the following levels the error occurred: • transport: secure transport • rpc: messages • protocol: operations • application: content

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

11.1.7 Network Management Datastore Architecture (NMDA)

The Network Management Datastore Architecture (NMDA), as specified in RFC 8342, provides the framework for NETCONF, where each datastore is exposed separately. RFC 8256 provides the extensions to the NETCONF protocol to support the NMDA.

The NMDA separates configuration datastores from datastores that contain the state of the device (such as the operational datastore). It also provides a framework where different datastores can be combined together to provide an operator views of configurations where translations, expansions and replacements are performed.

The following table provides an overview of the supported datastores.

Table 27: Supported datastores from the NDMA

Datastore	Description
running	This datastore contains the current configuration of the device. By default, changes committed to the running configuration datastore (using a `<commit>` operation from the candidate datastore) are not automatically copied to the startup configuration datastore; a `<copy-config>` operation is required to achieve this. This means that committed changes are not persistent after a reboot of the system unless the `<copy-config>` operation has been performed. If the `system configuration auto-save true` configuration has been applied, SR Linux automatically performs a `<copy-config>` operation between the running and startup datastores following a successful change to the running configuration datastore.
candidate	The candidate datastore, also known as a shared candidate datastore, can be manipulated without affecting the running configuration of the system. When the candidate configuration is ready, you can `<commit>` this configuration into the running datastore. Multiple NETCONF clients or CLI users can concurrently make changes to the same candidate datastore. A `<commit>` operation applies all configuration that exists in the candidate datastore no matter which user made the changes. The candidate configuration datastore is represented by the named configuration datastore called `default` in the SR Linux CLI.
startup	This datastore contains the configuration that the device uses at boot up. Changes made to other datastores are not automatically reflected in the startup datastore and an explicit `<copy-config>` operation is required. If the `system configuration auto-save true` configuration has been applied, SR Linux automatically performs a `<copy-config>` operation between the running and startup datastores following a successful change to the running configuration datastore. The startup configuration datastore obtains information from, and writes information to, the saved `/etc/opt/srlinux/config.json` configuration file. The NETCONF representation of this data is always provided in correctly namespaced XML format. The contents of this datastore is copied into the running datastore at system boot up.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Datastore	Description
	The configuration in the startup datastore may contain configuration that requires transformations before it can be applied to the system.
intended	This read-only datastore contains the complete configuration of the device, including all configuration transformations (in their fully expanded form).
operational	This datastore is a read-only datastore that contains all the state and configuration in use by the system. The configuration contained in the operational datastore is the fully expanded configuration that would be found in the intended datastore.

11.1.8 NETCONF server capabilities

Capabilities are advertised in the `<hello>` messages exchanged by NETCONF client and the SR Linux NETCONF server when a session is started.

Each peer (client and server) must send at least the base NETCONF capability, `urn:ietf:params:netconf:base:1.1`. SR Linux also sends the older version 1.0 base NETCONF capability `urn:ietf:params:netconf:base:1.0`.

Both NETCONF peers verify that the other peer has advertised a common base protocol version. When comparing protocol version capability URIs, if any parameters are encoded at the end of the URI string, they are ignored. If no base protocol version capability in common is found, the NETCONF session terminates. If more than one protocol version URI in common is present, both peers use the latest protocol version.

If the base protocol selected is `urn:ietf:params:netconf:base:1.0`, the standard framing method is used. This means that each NETCONF message sent and received is terminated with the following special set of characters:]]>]]>. While this method is sufficient for most NETCONF operations, in some cases, the]]>]]> string may occur naturally in a NETCONF message. In this very rare case, the NETCONF message may not be successfully received and parsed by the client or server.

Following is an example of a traditionally framed NETCONF transaction:

```
<rpc message-id="10" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <running/>
        </source>
        <filter>
            <network-instance xmlns="urn:nokia.com:srlinux:net-inst:network-instance"/>
        </filter>
    </get-config>
</rpc>
]]>]]>
<rpc-reply message-id="10" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <network-instance xmlns="urn:nokia.com:srlinux:net-inst:network-instance">
            <name>mgmt</name>
            <type>ip-vrf</type>
            <admin-state>enable</admin-state>
            <description>Management network instance</description>
            <interface>
                <name>mgmt0.0</name>
            </interface>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
            <protocols>
                <linux xmlns="urn:nokia.com:srlinux:linux:linux">
                    <import-routes>true</import-routes>
                    <export-routes>true</export-routes>
                    <export-neighbors>true</export-neighbors>
                </linux>
            </protocols>
        </network-instance>
    </data>
</rpc-reply>
]]>]]>
```

If the base protocol selected is `urn:ietf:params:netconf:base:1.1`, the more modern chunked framing method is used. When using the chunked framing method, each NETCONF message is sent and received alongside additional metadata describing the size of the message in octets. Using this method, a well-known message terminator is not required.

Following is an example of a chunked framed NETCONF transaction:

```
#248
<rpc message-id="10" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source><running/></source>
    <filter>
      <network-instance xmlns="urn:nokia.com:srlinux:net-inst:network-instance"/>
    </filter>
  </get-config>
</rpc>
##

#76
<rpc-reply message-id="10" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">

#11
    <data>

#143
        <network-instance xmlns="urn:nokia.com:srlinux:net-inst:network-instance">

#30
            <name>mgmt</name>

#32
            <type>ip-vrf</type>

#46
            <admin-state>enable</admin-state>

#67
            <description>Management network instance</description>

#24
            <interface>

#37
                <name>mgmt0.0</name>

#25
            </interface>

#24
            <protocols>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
#66
               <linux xmlns="urn:nokia.com:srlinux:linux:linux">

#56
                   <import-routes>true</import-routes>

#56
                   <export-routes>true</export-routes>

#62
                   <export-neighbors>true</export-neighbors>

#25
               </linux>

#25
           </protocols>

#28
       </network-instance>

#12
    </data>

#13
</rpc-reply>

##
```

Note: Examples provided in this documentation use the standard framing method as provided by the `urn:ietf:params:netconf:base:1.0` capability.

Capabilities take the form of a URI or URL and some optional parameters, for example:

```
<capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:yang-library:1.1?revision=2019-01-04&content-id=
22.10.1</capability>
<capability>urn:ietf:params:xml:ns:yang:ietf-origin?module=ietf-origin&revision=2018-02-14</
capability>
<capability>http://openconfig.net/yang/telemetry?module=openconfig-telemetry&revision=2018-08-
17&deviations=nokia-sr-openconfig-telemetry-deviations</capability>
```

Note: If a URL is provided, this URL does not need to be resolvable nor does it need to provide any data at the URL destination.

Capabilities can advertise functionality specified in RFC specifications, YANG modules available on the node along with potential deviations, and less commonly, proprietary information.

The SR Linux NETCONF server advertises the following capabilities:

Table 28: Summary of SR Linux server capabilities

Capability (shorthand)	Advertised capability/description	Specification
base:1.0	`urn:ietf:params:netconf:base:1.0` Support for the original NETCONF base specification, including the original `]]>]]>` message delimiter and standard framing.	RFC 4742, *Using the NETCONF Configuration Protocol over Secure SHell (SSH)*

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Capability (shorthand)	Advertised capability/description	Specification
base:1.1	`urn:ietf:params:netconf:base:1.1` Support for the current NETCONF base specification using chunked framing.	RFC 6242, *Using the NETCONF Protocol over Secure Shell (SSH)*
candidate	`urn:ietf:params:netconf:capability:candidate:1.0` Support for the <commit> and <discard-changes> NETCONF operations. Support for the candidate datastore as input for the <source> or <target> parameters for the following operations: • <get-config> • <edit-config> • <copy-config> • <validate> • <lock> • <unlock> Support for the candidate datastore as an input for the following NMDA-defined operations. • <get-data> • <edit-data> For more information about the startup datastore, see Network Management Datastore Architecture (NMDA).	RFC 6241, *Network Configuration Protocol (NETCONF)* RFC 8526, *NETCONF Extensions to Support the Network Management Datastore Architecture*
confirmed-commit	`urn:ietf:params:netconf:capability:confirmed-commit:1.1` Support for the <cancel-commit> NETCONF operation. Support for the following additional parameters for the <commit> operations: <confirmed>, <confirm-timeout>, <persist>, and <persist-id>. For details, see commit.	RFC 6241, *Network Configuration Protocol (NETCONF)*
rollback-on-error	`urn:ietf:params:netconf:capability:rollback-on-error:1.0` Supports the `rollback-on-error` option for the <error-option> parameter in the <edit-config> NETCONF operation. **Note:** The system rolls back all changes made in the shared candidate configuration no matter which user or session made them. Use of the <lock> operation when using `rollback-on-error` and the shared candidate configuration datastore is strongly recommended	RFC 6241, *Network Configuration Protocol (NETCONF)*

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Capability (shorthand)	Advertised capability/description	Specification
validate	`urn:ietf:params:netconf:capability:validate:1.1` Support for the <validate> NETCONF operation. For details, see validate. Also enables the use of the <test-option> parameter in the <edit-config> NETCONF operation.	RFC 6241, *Network Configuration Protocol (NETCONF)*
validate	`urn:ietf:params:netconf:capability:validate:1.0` Support for the original <validate> NETCONF operation. This capability is provided for backwards compatibility. As the server advertises both validate capabilities, clients are expected to use the 1.1 version.	RFC 4741, *NETCONF Configuration Protocol*
startup	`urn:ietf:params:netconf:capability:startup:1.0` Support for the startup configuration datastore. For more information about the startup datastore, see Network Management Datastore Architecture (NMDA). Support for the startup datastore as input for the <source> or <target> parameters in the following operations: • <get-config> • <copy-config> • <get-data> • <lock> • <unlock> • <validate> • <delete-config> Performing a <delete-config> operation on the startup datastore is the same as factory defaulting the routers configuration. SR Linux does not support the use of the startup datastore with the <edit-config> and <edit-data> operations.	RFC 6241, *Network Configuration Protocol (NETCONF)*
url	`urn:ietf:params:netconf:capability:url:1.0?scheme={http,https,ftp,sftp,file}` Provides the url option as a source or target for the following operations: • <edit-config> • <edit-data> • <copy-config> • <validate>	RFC 6241, *Network Configuration Protocol (NETCONF)*

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Capability (shorthand)	Advertised capability/description	Specification
	SR Linux does not support the `url` option for the `<delete-config>` operation. SR Linux support for HTTP, HTTPS, FTP, SFTP, and file URL formats.	
with-defaults	`urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=explicit&also-supported=report-all` Support for the `<with-defaults>` option in the following operations: • `<get>` • `<get-config>` • `<get-data>` • `<copy-config>` The SR Linux NETCONF server uses the `explicit` mode by default and supports `report-all` as an alternative. The `trim` mode is not supported on SR Linux. For related information, see with-defaults option.	RFC 6243, *With-defaults Capability for NETCONF*
with-operational-defaults	`urn:ietf:params:netconf:capability:with-operational-defaults:1.0` Support for the `<with-defaults>` option with the `<get-data>` operation targeting the operational datastore. The `with-operational-defaults` option is advertised in conjunction with the `with-defaults` capability. For related information, see with-defaults option.	RFC 8526, *NETCONF Extensions to Support the Network Management Datastore Architecture*
yang-library	`urn:ietf:params:netconf:capability:yang-library:1.1?revision=2019-01-04&content-id=<content-id-value>` Support for the IETF YANG library module and the Network Management Datastore Architecture (NMDA). For related information, see YANG library.	RFC 8525 *NETCONF Extensions to Support the Network Management Datastore Architecture* RFC 8526, *NETCONF Extensions to Support the Network Management Datastore Architecture*
ietf-netconf-monitoring	`urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring` Support for the IETF NETCONF monitoring information. For related information, see NETCONF IETF monitoring.	RFC 6022, *YANG Module for NETCONF Monitoring*

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

11.2 NETCONF configuration

This section describes how to configure the NETCONF server including configuring SSH transport, configuring the NETCONF server, adding filters, logging, and initiating a connection.

11.2.1 Configuring the SSH transport

NETCONF on SR Linux operates over the SSH transport. The SSH server in SR Linux is configured under the **system ssh-server** context.

Operators can choose whether to create a specific SSH server for the NETCONF protocol or whether to use an existing SSH server. The selected SSH server and its associated configuration determines the TCP port that the NETCONF protocol, which runs as a SSH subsystem, is bound to. A NETCONF server instance may bind to a single SSH server instance. Multiple NETCONF server instances may be created as required.

Cipher, message authentication code (MAC), key exchange, and host key algorithm lists are configured per SSH server instance, including SSH server instances used for NETCONF. For more information, see the "Configuring SSH cryptographic algorithms" section in the *SR Linux Configuration Basics Guide*.



Note: When an SSH server instance is created, users with the appropriate permissions (for example, admin and linuxadmin) will be able to access the CLI shell, unless the **disable-shell true** configuration is applied. This configuration restricts the specified SSH server on the corresponding TCP port to the associated SSH subsystems only (NETCONF) and removes the ability to access the CLI.

The following is an example configuration of an SSH server that will be used to support NETCONF. This example provides the following:

- an SSH server on the IANA defined TCP port for NETCONF of 830
- operation in the mgmt network-instance
- the restriction of shell access using this SSH server instance

Example

```
system {
    ssh-server mgmt-netconf {
        admin-state enable
        network-instance mgmt
        port 830
        disable-shell true
    }
}
```

11.2.2 Configuring the NETCONF server

Having ensured an SSH server instance is available and configured, the next step is to create a Authentication, Authorization and Accounting (AAA) role that specifically allows the NETCONF operations needed.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

The following example shows a AAA role configuration that enables specific NETCONF operations. Replace the <list-of-allowed-rpcs> field with one or more NETCONF operations as required. For details, see Enabling the NETCONF operations.

```
system {
    aaa {
        authorization {
            role netconf {
                services [
                    netconf
                ]
                netconf {
                    allowed-operations [ <list-of-allowed-rpcs...> ]
                }
            }
        }
    }
}
```

You can now configure a NETCONF server. The NETCONF server configuration creates a named netconf-server instance and binds it to a configured SSH server as described in Transport and sessions. NETCONF sessions over SSH count toward the system SSH connection limit, if configured. See Configuring the maximum number of SSH connections.

The following example configuration creates a NETCONF server called mgmt and binds it to the SSH server named mgmt-netconf:

```
system {
    ssh-server mgmt-netconf {
        admin-state enable
        network-instance mgmt
        port 830
        disable-shell true
    }
    netconf-server mgmt {
        admin-state enable
        ssh-server mgmt-netconf
    }
}
```

11.2.2.1 Enabling the NETCONF operations

The NETCONF server provides multiple operations that are disabled by default. The operations are enabled one-by-one in an AAA role configuration under /system aaa authorization role <rolename> netconf allowed-operations. The list of available operations is shown below:

- <cancel-commit>
- <close-session>
- <commit>
- <copy-config>
- <delete-config>
- <discard-changes>
- <edit-config>

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- `<edit-data>`
- `<get>`
- `<get-config>`
- `<get-data>`
- `<get-schema>`
- `<kill-session>`
- `<lock>`
- `<unlock>`
- `<validate>`

For details about the NETCONF operations, see NETCONF operations.

11.2.3 Adding filter entries to allow connections to the chosen TCP port

The factory installed CPM filters for IPv4 and IPv6 allow the default NETCONF TCP port to access the system. If this is not required, they should be removed.

If using an alternative TCP port for NETCONF or if defining custom CPM filters instead of the system default ones, add the following entries to the required `ipv4-filter` and the `ipv6-filter` CPM filters.

Example: IPv4-adding filter entries to allow connections to the default port specified by `<port-number>`

```
acl {
    acl-filter cpm type ipv4 {
        entry 400 {
            description "Accept incoming SSH connections on the NETCONF port"
            match {
                ipv4 {
                    protocol tcp
                }
                transport {
                    destination-port {
                        operator eq
                        value <port-number>
                    }
                }
            }
            action {
                accept {
                }
            }
        }
    }
}
```

Example: IPv6-adding filter entries to allow connections to the NETCONF port specified by the `<port-number>`

```
acl {
    acl-filter cpm type ipv6 {
        entry 440 {
            description "Accept incoming SSH connections on the NETCONF port"
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
            match {
                ipv6 {
                    next-header tcp
                }
                transport {
                    destination-port {
                        operator eq
                        value <port-number>
                    }
                }
            }
            action {
                 accept {
                 }
            }
        }
    }
}
```

11.2.4 UNIX socket

A **netconf-server** may be configured to use a UNIX socket using the **system netconf-server** *<name>* **unix-socket** command. More than one **netconf-server** instance with UNIX socket can be configured.

When a UNIX socket is successfully configured, a UNIX socket file is created at `/opt/srlinux/var/run` with the name `sr_netconf_server_<PID>`, where `PID` is the process ID of the running **netconf-server** instance.

If the UNIX socket cannot be created, the system reattempts the UNIX socket creation in the background. No action is required of the operator. If you do not want the system to reattempt UNIX socket creation in the background, you can either change the transport type or administratively disable the **netconf-server** instance.

When a UNIX socket cannot be created, the status is shown in the NETCONF server's **open-down-reason**. The following states can be displayed, depending on the situation:

- `admin-disabled`
- `ssh-server-down`
- `socket-create-failed`
- `socket-bind-failed`
- `socket-file-create-failed`
- `out-of-resources`
- `out-of-memory`

When reporting the state of a UNIX socket-based NETCONF session in the `ietf-netconf-monitoring.yang` YANG model, the **ietf-netconf-monitoring:netconf-state sessions session transport** leaf reports `netconf-ssh`, as no other appropriate identity value is defined in the IETF YANG model. The **ietf-netconf-monitoring:netconf-state sessions session source-host** leaf reports as `localhost`.

The following is an example of an RPC request/reply for the **netconf-state**.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: RPC request/reply for the netconf-state

```
<rpc message-id="10" 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>
]]>]]>
<rpc-reply message-id="10" 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>59</session-id>
                    <transport>netconf-ssh</transport>
                    <username>admin</username>
                    <source-host>localhost</source-host>
                    <login-time>2025-06-23T12:38:24.723Z</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>
]]>]]>
```

11.2.5 SR Linux logging

SR Linux performs logging for several operations.

Additional extensive logging can be enabled for debugging purposes using the `traceoptions` configuration container.

The following items are logged by default:

- Session start/stop
- Session client IP address
- Transport (SSH only currently)
- Each RPC start/stop/operation type/datastore targeted (if applicable)/success or failure/error on failure

UNIX socket create/fail scenarios produce the folloiwng logs by default:

- **netconf-server** instance socket creation success
- **netconf-server** instance socket creation error

With `trace-options` enabled, the system logs at the DEBUG level all of the preceding elements and in addition:

- the input content of each message in XML
- the output content of each message in XML

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Note: Nokia recommends that extensive logging for NETCONF using `trace-options` be used for debugging purposes only.

11.2.6 Initiating a connection to a NETCONF server

To connect to the NETCONF server on SR Linux, a NETCONF client uses the SSH protocol to connect to the SR Linux device on the chosen TCP port and requests the `netconf` SSH subsystem.

This can be done manually to test the server using the following command assuming:

- The SR Linux device is available on IP address 192.168.0.1.
- The admin user has the correct permissions to connect to the node.
- The SSH server for NETCONF is configured to run on TCP port 830.

Example: Initiating a NETCONF session

```
ssh -l admin -p 830 192.168.0.1 -s netconf
```

11.3 NETCONF state information

The SR Linux NETCONF implementation provides statistical counters on a per NETCONF session basis and aggregated for the NETCONF server instance. The statistics are available under the statistics container.

In addition, state information provides the operating state of the NETCONF server and the date and time of the last operational state change.

SR Linux native models

The following is an example of the state information in the SR Linux native models.

```
+--rw system
  +--rw netconf-server* [name] {srl-feat:netconf}?
     +--ro oper-state?          srl-comm:oper-state
     +--ro last-oper-change?    srl-comm:date-and-time
     +--ro statistics
        +--ro active-sessions?                     uint8
        +--ro total-requests?                      srl-comm:zero-based-counter32
        +--ro total-responses?                     srl-comm:zero-based-counter32
        +--ro total-error-responses?               srl-comm:zero-based-counter32
        +--ro total-close-session-requests?        srl-comm:zero-based-counter32
        +--ro total-in-bad-hellos?                 srl-comm:zero-based-counter32
        +--ro total-dropped-sessions?              srl-comm:zero-based-counter32
        +--ro total-get-requests?                  srl-comm:zero-based-counter32
        +--ro total-get-config-requests?           srl-comm:zero-based-counter32
        +--ro total-get-data-requests?             srl-comm:zero-based-counter32
        +--ro total-get-schema-requests?           srl-comm:zero-based-counter32
        +--ro total-edit-config-requests?          srl-comm:zero-based-counter32
        +--ro total-failed-edit-config-requests?   srl-comm:zero-based-counter32
        +--ro total-edit-data-requests?            srl-comm:zero-based-counter32
        +--ro total-failed-edit-data-requests?     srl-comm:zero-based-counter32
        +--ro total-kill-session-requests?         srl-comm:zero-based-counter32
        +--ro total-copy-config-requests?          srl-comm:zero-based-counter32
        +--ro total-delete-config-requests?        srl-comm:zero-based-counter32
        +--ro total-validate-requests?             srl-comm:zero-based-counter32
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
        +--ro total-lock-requests?                 srl-comm:zero-based-counter32
        +--ro total-failed-lock-requests?          srl-comm:zero-based-counter32
        +--ro total-unlock-requests?               srl-comm:zero-based-counter32
        +--ro total-commit-requests?               srl-comm:zero-based-counter32
        +--ro total-discard-changes-requests?      srl-comm:zero-based-counter32
        +--ro total-action-requests?               srl-comm:zero-based-counter32
        +--ro session* [session-id]
           +--ro session-id                     uint32
           +--ro process-id?                    uint32
           +--ro in-bad-hellos?                 srl-comm:zero-based-counter32
           +--ro get-requests?                  srl-comm:zero-based-counter32
           +--ro get-config-requests?           srl-comm:zero-based-counter32
           +--ro get-data-requests?             srl-comm:zero-based-counter32
           +--ro get-schema-requests?           srl-comm:zero-based-counter32
           +--ro edit-config-requests?          srl-comm:zero-based-counter32
           +--ro failed-edit-config-requests?   srl-comm:zero-based-counter32
           +--ro edit-data-requests?            srl-comm:zero-based-counter32
           +--ro failed-edit-data-requests?     srl-comm:zero-based-counter32
           +--ro kill-session-requests?         srl-comm:zero-based-counter32
           +--ro copy-config-requests?          srl-comm:zero-based-counter32
           +--ro delete-config-requests?        srl-comm:zero-based-counter32
           +--ro validate-requests?             srl-comm:zero-based-counter32
           +--ro lock-requests?                 srl-comm:zero-based-counter32
           +--ro failed-lock-requests?          srl-comm:zero-based-counter32
           +--ro unlock-requests?               srl-comm:zero-based-counter32
           +--ro commit-requests?               srl-comm:zero-based-counter32
           +--ro discard-changes-requests?      srl-comm:zero-based-counter32
           +--ro action-requests?               srl-comm:zero-based-counter32
```

IETF standard models

NETCONF supports standardized IETF YANG models that contain state data relating explicitly to the NETCONF server. These IETF models (along with OpenConfig models) are returned by the various NETCONF operations (unless specifically filtered out). While these IETF modules are visible in the native schema management server in CLI and gNMI, you cannot obtain data from them using the CLI or gNMI.

11.4 NETCONF operations

The SR Linux NETCONF implementation supports the following operations:

- get
- get-config
- get-data
- get-schema
- edit-config
- edit-data
- copy-config
- discard-changes
- commit
- cancel-commit
- lock

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- unlock
- delete-config
- close-session
- validate
- action
- kill-session

Where appropriate, Nokia recommends the use of the NDMA <get-data> and <get-data> operations for obtaining information and configuring the device.

Operations and mixed module author groups

SR Linux supports mixed module author groups, such as SR Linux native YANG modules. In conjunction with OpenConfig YANG modules in the current implementation of NETCONF in SR Linux:

- NETCONF clients can make <edit-config> and <edit-data> requests that target Nokia-authored SR Linux native YANG models and OpenConfig authored YANG models.
  For example, if you make an<edit-config> request in the SR Linux native YANG first and then make an <edit-config> request using the OpenConfig YANG for the same leaf, the SR Linux configuration will apply (even though the OpenConfig setting was the last write). The same rule applies when an <edit-config> request includes both models in a single transaction.
- A <commit> operation applies the changes to both running configuration datastores (that is, OpenConfig and SR Linux). Both candidates are cleared of configuration, returning the operator to the beginning where another choice of model author group in the candidate configuration may be made.
- The current implementation supports requests to obtain state data from the running configuration using the <get> RPC or from the operational datastore using the <get-data> RPC filtered on either or both models.
- Performing a <lock> on the candidate datastore locks both the OpenConfig and the native SR Linux management server default shared candidate configurations, making the candidate datastore exclusive.
- Requests to obtain state data from the running configuration datastore using the <get> RPC or from the operational datastore using the <get-data> RPC that do not contain a filter returns data represented in both SR Linux native and OpenConfig YANG modules (if they are enabled in the system).

11.4.1 get

The <get> operation, as defined in RFC 6241, retrieves configuration and state data from the running configuration datastore.

Table 29: Parameters for the <get> operation

Parameter	Description	Comment
<filter>	Limits the data returned. The filter provided must be a correctly namespaced filter restricting the data returned from configuration, state, or both.	Optional. If no filter is specified, the operation returns all configuration and state

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Parameter	Description	Comment
	SR Linux supports only the subtree filter type. For details, see NETCONF filtering.	information from the running datastore.
<with-defaults>	Specifies how the NETCONF server handles the retrieval of default data. For details, see with-defaults option.	Optional.

In a successful RPC, the response includes a <data> node that contains the requested data. In an unsuccessful RPC, the response includes the <rpc-error>.

Example: RPC request and reply messages for the get operation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get>
        <filter>
            <system xmlns="urn:nokia.com:srlinux:general:system">
                <ssh-server xmlns="urn:nokia.com:srlinux:linux:ssh">
                    <name>mgmt</name>
                </ssh-server>
            </system>
        </filter>
    </get>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <system xmlns="urn:nokia.com:srlinux:general:system">
            <ssh-server xmlns="urn:nokia.com:srlinux:linux:ssh">
                <name>mgmt</name>
                <admin-state>enable</admin-state>
                <oper-state>up</oper-state>
                <network-instance>mgmt</network-instance>
                <port>22</port>
                <disable-shell>false</disable-shell>
                <timeout>0</timeout>
                <rate-limit>20</rate-limit>
                <protocol-version>2</protocol-version>
                <host-key>
                    <preserve>true</preserve>
                </host-key>
            </ssh-server>
        </system>
    </data>
</rpc-reply>
]]>]]>
```

11.4.2 get-config

The <get-config> operation retrieves data from the running or candidate configuration datastores.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Table 30: Parameters for the <get-config> operation

Parameter	Description	Comment
<source>	Specifies the configuration datastore to be queried. Valid options: • <running/> • <candidate/> • <startup/>	Mandatory
<filter>	Limits the data returned. SR Linux supports the subtree filter type. For related information, see NETCONF filtering.	Optional If no filter is specified, the response includes all configuration and state data from the specified datastore.
<with-defaults>	Specifies how the NETCONF server handles the retrieval of default data. Supported options are explicit and report-all. For details, see with-defaults option.	Optional

Example: RPC request and reply messages for the <get-config> operation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <candidate/>
        </source>
        <filter>
            <system xmlns="urn:nokia.com:srlinux:general:system">
                <ssh-server xmlns="urn:nokia.com:srlinux:linux:ssh">
                    <name>mgmt</name>
                </ssh-server>
            </system>
        </filter>
    </get-config>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <system xmlns="urn:nokia.com:srlinux:general:system">
            <ssh-server xmlns="urn:nokia.com:srlinux:linux:ssh">
                <name>mgmt</name>
                <admin-state>enable</admin-state>
                <network-instance>mgmt</network-instance>
            </ssh-server>
        </system>
    </data>
</rpc-reply>
]]>]]>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

11.4.3 get-data

The <get-data> operation is the NMDA version of the <get> operation.

The <get-data> operation is defined in the ietf-netconf-nmda YANG module and exists in the xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda" namespace, which must be correctly provided to the <get-data> operation.

Table 31: Parameters for the <get-data> operation

Parameter	Description	Comment
<datastore>	Identifies the datastore to be queried. The datastore is an identity reference (identityref) located in the ietf-datastores (or any other module that augments itself into this field) YANG module in the namespace urn:ietf:params:xml:ns:yang:ietf-datastores. As these are identityrefs, the value must be provided with a namespace prefix declaration xmlns:ds= "urn:ietf:params:xml:ns:yang:ietf-datastores" and this declared namespace must be used in the value of the datastore field, for example ds:candidate, where ds is the declared XML namespace prefix. Available options: • ds:running • ds:candidate • ds:intended • ds:startup • ds:operational	Mandatory No default The with-defaults and with-origin options are supported only with the operational datastore.
<subtree-filter>	Limits the returned data in subtree filter format. The filter provided must be a correctly namespaced filter restricting the data returned from the server. For details, see Subtree filtering.	Optional By default, all data from the specified datastore is returned.
<config-filter>	This is a boolean parameter. When <config-filter> is set to true, only YANG nodes set to config true will be returned. When <config-filter> is set to false, only YANG nodes set to config false will be returned.	Optional By default, all data from the specified datastore is returned.
<origin-filter>	The <origin-filter> parameter limits returned data to nodes that have a specified origin	Optional

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Parameter	Description	Comment
	(correctly namespaced using the `xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"` namespace declaration). Multiple <origin-filter> elements can be specified. When more than one filter exists, <origin-filter> is treated as an OR; that is, the XML node is returned if it matches any of the specified <origin-filter> statements. Any XML node that does not have an <origin> set is treated as though it has an <origin> equal to unknown. The <origin-filter> does not affect state data which is returned regardless of the <origin-filter>. If state data is not required, it can be limited using the <config-filter> parameter. The <origin-filter> parameter is mutually exclusive with the <negated-origin-filter> parameter. See <origin-filter> parameter configuration for an example <origin-filter> configuration.	By default, all data from the specified datastore is returned.
<negated-origin-filter>	The <negated-origin-filter> parameter limits the returned data to nodes that do not have a specified origin (correctly namespaced using the `xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"` namespace declaration). Multiple <negated-origin-filter> parameters can be configured. When more than one filter exists, <negated-origin-filter> is treated as a NOR; that is, the XML node is returned only if it does not match any of the specified <negated-origin-filter> statements. Any XML node that does not have an <origin> set is treated as though it has <origin> equal to unknown . The <negated-origin-filter> parameter does not affect state data that is returned regardless of the <negated-origin-filter>. If state data is not required, this can be limited using the <config-filter> parameter. The <negated-origin-filter> parameter is mutually exclusive with the <origin-filter> parameter.	Optional By default, all data from the specified datastore is returned.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Parameter	Description	Comment
<max-depth>	Restricts the data returned to the provided maximum number of subtrees.	Optional By default, all data from the specified datastore is returned.
<with-origin>	Outputs the origin information for each returned node (except non-presence containers and config false elements). The origin information is correctly namespaced using the xmlns:or= "urn:ietf:params:xml:ns:yang:ietf-origin" declaration, if datastore is set to operational. If datastore is not set to operational, an rpc-error is reported with the error-tag set to invalid-value.	Optional By default, all data from the specified datastore is returned without any origin information displayed.
<with-defaults>	For the operational datastore, if no with-defaults parameter value is specified, or if it is set to explicit or report-all, the "in use" values, as defined in the operational datastore origin section above, are returned from operational state, even if a node has a default statement in the YANG module and the server is using this default value.	Optional

The following example shows an RPC request and reply message for the <get-data> operation performed against the operational datastore, with the with-origin and with-defaults arguments configured.

Example

```
<rpc message-id="101" 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-defaults xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">report-all</
with-defaults>
        <with-origin/>
        <subtree-filter>
            <system xmlns="urn:nokia.com:srlinux:general:system">
                <ssh-server xmlns="urn:nokia.com:srlinux:linux:ssh">
                    <name>mgmt</name>
                </ssh-server>
            </system>
        </subtree-filter>
    </get-data>
</rpc>
]]>]]>

<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-nmda">
        <system xmlns="urn:nokia.com:srlinux:general:system" xmlns:or=
"urn:ietf:params:xml:ns:yang:ietf-origin">
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
            <ssh-server xmlns="urn:nokia.com:srlinux:linux:ssh" or:origin="or:intended">
                <name>mgmt</name>
                <admin-state>enable</admin-state>
                <oper-state>up</oper-state>
                <network-instance>mgmt</network-instance>
                <port>22</port>
                <disable-shell>false</disable-shell>
                <timeout>0</timeout>
                <rate-limit>20</rate-limit>
                <protocol-version>2</protocol-version>
                <host-key>
                    <preserve>true</preserve>
                </host-key>
            </ssh-server>
        </system>
    </data>
</rpc-reply>
]]>]]>
```

Example: <origin-filter> parameter configuration

```
<rpc message-id="102" 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" xmlns:or="urn:ietf:params:xml:ns:yang:ietf-
origin">
        <datastore>ds:operational</datastore>
        <subtree-filter>
            <bgp xmlns="http://example.com/ns/bgp"/>
        </subtree-filter>
        <origin-filter>or:intended</origin-filter>
        <origin-filter>or:system</origin-filter>
        <with-origin/>
    </get-data>
</rpc>

<rpc-reply message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
        <bgp xmlns="http://example.com/ns/bgp" xmlns:or="urn:ietf:params:xml:ns:yang:ietf-
origin" or:origin="or:intended">
            <peer>
                <name>2001:db8::2:3</name>
                <local-port or:origin="or:system">60794</local-port>
                <state>established</state>
            </peer>
        </bgp>
    </data>
</rpc-reply
```

11.4.4 get-schema

The <get-schema> operation is defined in the IETF NETCONF monitoring specifications, as described in RFC 6022.

The SR Linux NETCONF server advertises the urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring capability for the <get-schema> operation to be available.

The <get-schema> operation allows the NETCONF server to provide the requested YANG modules to the NETCONF client. Management systems typically use this operation to build the correct YANG schema for the device using the set of YANG modules advertised by the node. The <get-schema> operation

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

serves a single YANG module to the client. NETCONF clients typically request all YANG modules that they require. For related information, see NETCONF IETF monitoring.

Table 32: Parameters for the <get-schema> operation

Parameter	Description	Comment
<identifier>	Specifies the name of the YANG module. This value must match exactly any module name or identifier described by the node in any of the IETF YANG library or IETF NETCONF monitoring trees.	Mandatory No default value
<version>	Specifies the version and revision of the YANG module. If provided, the value must match exactly any version or revision described by the node.	Optional No default
<format>	Specifies the format of the YANG schema output. SR Linux only supports the yang format.	Optional If format is specified, the value is always yang.

If the operation is successful, the RPC reply includes a <data> node containing the requested YANG module.

If the requested YANG module does not exist, the <error-tag> field is set to invalid-value.

If more than one YANG module matches the requested parameters, the <error-tag> field reports operation-failed and the <error-app-tag> field is set to data-not-unique.

Example: RPC request and response for a <get-schema> operation

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <get-schema xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
      <identifier>ietf-datastores</identifier>
    </get-schema>
</rpc>
]]>]]>

<rpc-reply message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring"><![CDATA[module
 ietf-datastores {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-datastores";
  prefix ds;

  organization
    "IETF Network Modeling (NETMOD) Working Group";

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>

     WG List:  <mailto:netmod@ietf.org>

     Author:   Martin Bjorklund
               <mailto:mbj@tail-f.com>

     Author:   Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>

     Author:   Phil Shafer
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
              <mailto:phil@juniper.net>

   Author:    Kent Watsen
              <mailto:kwatsen@juniper.net>

   Author:    Rob Wilton
              <rwilton@cisco.com>";

 description
   "This YANG module defines a set of identities for identifying
    datastores.

    Copyright (c) 2018 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject to
    the license terms contained in, the Simplified BSD License set
    forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (https://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC 8342
    (https://www.rfc-editor.org/info/rfc8342); see the RFC itself
    for full legal notices.";

 revision 2018-02-14 {
   description
     "Initial revision.";
   reference
     "RFC 8342: Network Management Datastore Architecture (NMDA)";
 }

 /*
  * Identities
  */

 identity datastore {
   description
     "Abstract base identity for datastore identities.";
 }

 identity conventional {
   base datastore;
   description
     "Abstract base identity for conventional configuration
      datastores.";
 }

 identity running {
   base conventional;
   description
     "The running configuration datastore.";
 }

 identity candidate {
   base conventional;
   description
     "The candidate configuration datastore.";
 }

 identity startup {
   base conventional;
   description
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
      "The startup configuration datastore.";
  }

  identity intended {
    base conventional;
    description
      "The intended configuration datastore.";
  }

  identity dynamic {
    base datastore;
    description
      "Abstract base identity for dynamic configuration datastores.";
  }

  identity operational {
    base datastore;
    description
      "The operational state datastore.";
  }

  /*
   * Type definitions
   */

  typedef datastore-ref {
    type identityref {
      base datastore;
    }
    description
      "A datastore identity reference.";
  }
}
]]></data>
</rpc-reply>
]]>]]>
```

11.4.5 edit-config

The <edit-config> operation is used to change the configuration of the SR Linux device. You can use it to make full or partial changes to configurations from a local file, remote file, or in-line to the node.

The SR Linux implementation of the <edit-config> operation has the following characteristics:

- A single <edit-config> operation can include multiple configuration changes. The changes can be made in and from multiple YANG modules (whether augmented or top-level). For additional restrictions, see Operations and mixed module author groups.
- An <edit-config> operation is atomic, meaning that all of the configuration applies or none of it does.
- Within a single transaction, the order of the data is not important with the exception of leaf-lists and user-ordered lists. SR Linux will reorder the provided data as required for the system.
- SR Linux supports configuration from URLs and advertises the :url capability. The <url> parameter can be used instead of the <config> parameter.

Each XML node in the specified configuration can have an optional operation attribute that defines the behavior of the configuration change. For supported options, see Options for the operation attribute. If the

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

operation attribute is not present, the NETCONF server uses the option specified in the <default-operation> parameter. If the <default-operation> parameter absent, the merge option is used.

Note: The <default-operation> and operation parameters have subtle but important differences in behaviour.

Table 33: Parameters for the edit-config operation

Parameter	Description	Comments
<target>	Specifies the configuration datastore being edited. SR Linux supports only the candidate datastore as the target for an <edit-config> operation.	Mandatory No default value
<default-operation>	Specifies the default operation for the configuration change. **Note:** The options available as the default-operation are more limited than on a per XML node basis and have slightly different behaviors. Supported values: • merge (the default): Merge the configuration data in the <config> or <url> parameter with the configuration at the corresponding level in the target datastore. This behavior is the default. • replace: Completely replaces the entire configuration data in the target datastore with that in the <config> or <url> parameters. This setting is useful for loading previously saved configuration data. • none: The target datastore is unaffected by the configuration in the <config> parameter, unless and until the incoming configuration data uses the operation attribute to request a different operation. If the configuration in the <config> or <url> parameter contains data for there is no corresponding level in the target datastore, the RPC error message includes an <error-tag> with a value of data-missing. The none option allows operations like delete to avoid unintentionally creating the parent hierarchy of the element to be edited.	Optional merge
<test-option>	Select from the following options: • test-then-set: Perform a validation test before attempting to set. If validation errors occur, do not perform the <edit-config> operation. This test option is the default.	Optional Default: test-then-set

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Parameter	Description	Comments
	• `test-only`: Perform only the validation test, without attempting to set. SR Linux does not support the `set` option where a test is performed without a validation test first.	
`<error-option>`	Specifies how errors are handled for the `<edit-config>` operation: • `stop-on-error`: Abort the `<edit-config>` operation when the first error occurs. While this option is the default in the NETCONF specification, it results in a partial configuration change (instead of an atomic change). While this option is accepted without error, it produces the same results as the `rollback-on-error` option on SR Linux. This option is provided, instead of deviating `stop-on-error` as not supported, to support clients that hard-code the `stop-on-error` setting. • `rollback-on-error`: If an error condition occurs and an `error` severity `<rpc-error>` element is generated, the server stops processing the `<edit-config>` operation and restores the specified configuration to its complete state at the start of this `<edit-config>` operation. A `warning` severity issue results in the `<edit-config>` succeeding. This behavior is the default in SR Linux. The `:validate:1.1` capability is advertised for the `rollback-on-error` option to work. **Note:** `continue-on-error` is not supported on SR Linux.	Optional `rollback-on-error` (as opposed to the specifications choice of `stop-on-error`)
`<config>`	Provides the configuration to be applied to the system. It is formatted as the complete hierarchy of configuration data as defined by the device's data models. The contents must be correctly namespaced, to allow the device to detect the appropriate data model, and the contents must follow the constraints of that data model, as defined by its capability definition. The value is encoded in YANG as `anyxml`. The `<config>` and `<url>` parameters are mutually exclusive.	No default

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Parameter	Description	Comments
<url>	The URL to a local or remote file that contains a complete hierarchy of configuration data to be applied as defined by the device's data models. The file is written in XML format. The contents must be correctly namespaced, to allow the device to detect the appropriate data model and the contents must follow the constraints of that data model, as defined by its capability definition. URL types that SR Linux supports are advertised in the :url capability. The value is encoded in YANG as anyxml. The <config> and <url> parameters are mutually exclusive.	No default

Example: RPC request and reply for the <edit-config> operation

```
<rpc message-id="10" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
        <target>
            <candidate/>
        </target>
        <config>
            <acl xmlns="urn:nokia.com:srlinux:acl:acl">
                <acl-filter>
                    <name>1</name>
                    <type>ipv4</type>
                </acl-filter>
            </acl>
        </config>
    </edit-config>
</rpc>
]]>]]>

<rpc-reply message-id="10" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

Related topics

Options for the operation attribute

11.4.6 edit-data

The <edit-data> operation changes the configuration of a datastore. It is similar to the <edit-config> operation, except it is designed as part of the NMDA specification to be extensible for any writeable NMDA datastore.

The <edit-data> operation differs from the <edit-config> operation in the following areas:

- The <test-option> parameter from the <edit-config> operation is not supported. The behavior is that of the default test-then-set approach of <edit-config> operation.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- The <error-option> parameter from <edit-config> is not supported. The behavior is that of rollback-on-error option in <edit-config> operation.

Each XML node in the specified configuration can have an optional operation attribute that defines the behavior of the configuration change. For supported options, see Options for the operation attribute. If the operation attribute is not present, the NETCONF server uses the option specified in the <default-operation> parameter. If the <default-operation> parameter absent, the merge option is used.

Note: The <default-operation> and operation parameters have subtle but important differences in behaviour.

Nokia recommends using the <edit-data> operation instead of the <edit-config> operation.

Table 34: Parameters for the <edit-data> operation

Parameter	Description	Comment
<datastore>	Specifies the datastore to be targeted; for SR Linux, the only option is ds:candidate. The datastore is an identity reference (identityref) located in the ietf-datastores (or any other module that augments itself into this field) YANG module in the urn:ietf:params:xml:ns:yang:ietf-datastores namespace. As these are identityrefs, the value must be provided with a namespace declaration xmlns:ds= "urn:ietf:params:xml:ns:yang:ietf-datastores" and this declared namespace must be used in the value of the datastore field, for example, ds:candidate where ds is the declared XML namespace.	Mandatory No default value.
<default-operation>	Specifies the default operation of the configuration change. **Note:** The options available for the default-operation option are more limited than on a per XML node basis and have slightly different behaviors. Supported values: • merge (the default): Merge the configuration data in the <config> or <url> parameters with the configuration at the corresponding level in the target datastore. This behavior is the default. • replace: Completely replace the entire configuration data in the target datastore with that in the <config> or <url> parameters. This setting is useful for loading previously saved configuration data.	Optional merge

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Parameter	Description	Comment
	• none: The target datastore is unaffected by the configuration in the <config> parameter, unless and until the incoming configuration data uses the operation attribute to request a different operation. If the configuration in the <config> or <url> parameter contains data for which there is no corresponding level in the target datastore, the RPC error message includes an <error-tag> with a value of data-missing. The none option allows operations such as <delete> to avoid unintentionally creating the parent hierarchy of the element to be edited.	
<config>	Provides the configuration to be applied to the system. It is formatted as the complete hierarchy of configuration data as defined by the device's data models. The contents must be correctly namespaced, to allow the device to detect the appropriate data model, and the contents must follow the constraints of that data model, as defined by its capability definition. The value is encoded in YANG using the YANG 1.1 anydata type. The <config> and <url> parameters are mutually exclusive.	No default
<url>	The URL to a local or remote file that contains a complete hierarchy of configuration data to be applied as defined by the device's data models. The file is written in XML. The contents must be correctly namespaced, to allow the device to detect the appropriate data model, and the contents must follow the constraints of that data model, as defined by its capability definition. The value is esncoded in YANG using the YANG 1.1 anydata type. URL types that SR Linux supports are advertised in the :url capability. The <config> and <url> parameters are mutually exclusive.	No default

Example: RPC request and reply messages for the <edit-data> operation

```
<rpc message-id="10" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda" xmlns:ds=
"urn:ietf:params:xml:ns:yang:ietf-datastores">
        <datastore>ds:candidate</datastore>
        <config>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
            <acl xmlns="urn:nokia.com:srlinux:acl:acl">
                <acl-filter>
                    <name>1</name>
                    <type>ipv4</type>
                </acl-filter>
              </acl>
        </config>
    </edit-data>
</rpc>
]]>]]>

<rpc-reply message-id="10" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

11.4.7 copy-config

Use the `<copy-config>` operation to create or replace the entire configuration datastore with that of another datastore (a datastore in this situation may be a file stored at a local/remote URL). If the destination (`target`) configuration datastore already exists, it is overwritten completely. If the target configuration datastore does not exist, it is created.

SR Linux does not support the following uses of the `<copy-config>` operation and returns an `<rpc-error>` with the `error-tag` set to `invalid-value`:

- using the running configuration datastore as a destination (`target`) for the copy
- performing remote-to-remote copy operations where the node itself is not the source or destination of the copy
- performing URL-to-URL copy operations (either local-to-remote, remote-to-local, or local-to-local)
- using the identical source and destination (`target`)
- using formats other than XML for the `<url>` of a `<source>` or `<target>` file

Performing a `<copy-config>` operation from the running configuration datastore to the `startup` configuration datastore is the same as saving the configuration.

Note: For configuration changes to persist after a reboot or switchover, after each `<commit>` operation, perform a `<copy-config>` operation with the `<source>` set to `<running/>` and the `<target>` set to `<startup/>`. If the `/system configuration auto-save true` configuration has been applied, SR Linux will automatically perform a `<copy-config>` operation between the running and startup datastores following a successful change to the running configuration datastore.

Table 35: Parameters for a `<copy-config>` operation

Parameter	Description	Comment
`<source>`	The source configuration datastore. Supported options are: • `<candidate/>` – the candidate configuration datastore • `<startup/>` – the startup configuration datastore	Mandatory No default value

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Parameter	Description	Comment
	• <url> – URL of the file to be replaced or created. This file may be a local or remote. You must have the correct permissions to update or create the target file The data read from the file is the entire contents of the required configuration datastore, encoded in XML format. For supported <target> and <source> combinations, see Supported source and target combinations.	
<target>	The target configuration datastore. Supported options are: <candidate/>: the candidate configuration datastore. This option does not commit the candidate configuration; a separate <commit> operation is required. <startup/>: the startup configuration datastore. <url>: URL of the file to be replaced or created. This file may be a local or remote. You must have the correct permissions to update or create the target file. The data written to the file is the entire contents of the <source> configuration datastore, encoded in XML format. For supported <target> and <source> combinations, see Supported source and target combinations.	Mandatory No default value
<with-defaults>	See with-defaults option. The <with-defaults> parameter is only applicable when the <target> of a <copy-config> operation is a URL, otherwise it is silently ignored.	Optional

Supported source and target combinations

If you specify source and target combinations that are not supported, the system reports an error message that describes the error.

Table 36: Source and target combinations

Source	Target	Supported
<candidate/>	<candidate/>	No
<candidate/>	<startup/>	No
<candidate/>	<running/>	No

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Source	Target	Supported
<candidate/>	<url>	Yes
<startup/>	<candidate/>	Yes
<startup/>	<startup/>	No
<startup/>	<running/>	No
<startup/>	<url>	Yes
<running/>	<candidate/>	No
<running/>	<startup/>	Yes
<running/>	<running/>	No
<running/>	<url>	Yes
<url>	<candidate/>	Yes
<url>	<startup/>	Yes
<url>	<running/>	No
<url>	<url>	No

Example: RPC request and reply message for the <copy-config> operation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <copy-config>
        <source>
            <url>https://user:password@example.com/cfg/new.xml</url>
        </source>
        <target>
            <candidate/>
        </target>
    </copy-config>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

11.4.8 discard-changes

The <discard-changes> operation, as described in RFC 6241, discards all changes in the shared candidate configuration datastore. This operation discards all changes made in the shared candidate configuration datastore, not just those changes made by the current NETCONF session.

Note: The <discard-changes> operation may also discard changes performed by users in another interface such as CLI or gNMI.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

This operation reverts the candidate configuration datastore to the contents of the current running configuration datastore.

Example: RPC request and reply for the <discard-changes> operation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <discard-changes/>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
]]>]]>
```

11.4.9 commit

The <commit> operation copies the configuration from the candidate datastore to the running datastore, overwriting anything previously in it. Additionally, the <commit> operation allows the configuration to be rolled back (automatically or manually) when used in conjunction with the <confirmed> parameter.

Note: Issuing a successful <commit> operation activates the configuration.

If the configuration cannot be committed, the commit operation fails and the running configuration datastore remains in the same state that it was in before the initiating the commit operation. The running configuration remains unchanged and the configuration integrity is ensured.

If the running configuration datastore is locked when the <commit> operation is issued, the operation fails with an <rpc-error> containing the <error-tag> with value of in-use.

An implicit lock is placed on the running configuration datastore for the duration of the <commit> operation. After a <commit> is successfully completed, confirmed, rolled back, or cancelled using the <cancel-commit> operation, the implicit lock on the running datastore is released.

Note: For configuration changes to persist after a reboot or switchover, after each <commit> operation, a <copy-config> operation should be performed with the <source> set to <running/> and the <target> set to <startup/>. If the /system configuration auto-save true configuration has been applied, SR Linux will automatically perform a <copy-config> operation between the running and startup datastores following a successful change to the running configuration datastore.

Table 37: Parameters for a commit operation

Parameter	Description	Comment
<confirmed>	Allows you to review and verify the configuration on the router before confirming (or cancelling) the committed changes. If the commit is not confirmed within the confirm-timeout duration, the changes are automatically rolled back and the candidate configuration is reinstated with the proposed changes.	Optional

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Parameter	Description	Comment
	If the <confirmed> parameter is absent, the commit takes effect immediately.	
<confirm-timeout>	Specifies the time in seconds to wait for the commit to be confirmed before automatically rolling back the changes. By default, the confirmed commit timeout is 600 seconds (10 minutes).	Optional
<persist>	Specifies a client-provided ID, a string of client-created characters. This ID is used to confirm the commit from another session. The <persist-id> and <persist> parameters are mutually exclusive.	Optional
<persist-id>	Specifies a previously chosen persist ID. This parameter is used to confirm a commit from another NETCONF session. If the value provided does not match a previously provided ID, the NETCONF operation fails with an invalid-value <rpc-error>. The <persist-id> and <persist> parameters are mutually exclusive.	Optional

Note: For information about how to cancel an ongoing confirmed commit, see cancel-commit.

The commit can be one for the following types:

- Immediate commit: commit the configuration from the candidate datastore without waiting for confirmation.
  Use the <commit/> tag without the confirmed parameter to commit without confirmation.
- Confirmed commit: wait for confirmation before committing the candidate to the running datastore. Use the confirmed parameter and optionally, confirm-timeout:

```
<commit>
    <confirmed/>
</commit>
```

or

```
<commit>
    <confirmed/>
    <confirm-timeout>30</confirm-timeout>
</commit>
```

If the <commit> is not confirmed within the timer, the original contents of the running datastore (before the commit) are restored into the running datastore and the proposed configuration changes remain in the candidate datastore. If the configuration changes disconnect the session and the <persist> option has not been set, the only way to revert them is to wait for the timer to expire. To confirm confirmed-commit, issue another <commit/> operation from the same session.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- Persistent confirmed commit: similar to a confirmed commit, except, if the <persist> parameter is present, you can confirm the commit using the commit operation from the current session or using the <persist-id> value from another session. If the configuration changes disconnect the session, you can revert them using the <persist-id> from another session.

Example: RPC request and reply for a persistent <commit> operation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <commit>
        <confirmed/>
        <persist>IQd4668</persist>
    </commit>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>

<!-- confirm the persistent confirmed-commit, possibly from another session -->
<rpc message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <commit>
        <persist-id>IQd4668</persist-id>
    </commit>
</rpc>
]]>]]>

<rpc-reply message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

11.4.10 cancel-commit

The <cancel-commit> operation cancels an ongoing confirmed commit.

Table 38: Parameters for a <cancel-commit> operation

Parameter	Description	Comment
<persist-id>	Use this parameter to cancel a commit from another NETCONF session. If the <persist-id> parameter is absent, the <cancel-commit> operation applies to the same session that issued the confirmed commit.	Optional

Example: RPC request and reply for a <cancel-commit> operation

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <cancel-commit/>
</rpc>
]]>]]>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

11.4.11 lock

The <lock> operation locks the configuration datastore and restricts other users or sessions from writing to it. Locking the running configuration datastore restricts other users or sessions from issuing a <commit> operation.

When the <lock> operation locks the shared candidate configuration datastore, no other session is allowed to use the <edit-config> operation. No other interface (including CLI, gNMI, or SNMP) can make changes to the locked configuration datastore. A lock on a configuration datastore remains in place until either it is unlocked using the <unlock> operation or until the NETCONF session closes (irrespective of whether the client or server closed the session).

While the <lock> operation explicitly locks the targeted configuration datastore, a configuration datastore may also be implicitly locked by the system. This occurs during a <commit> operation where SR Linux locks the running configuration datastore (if it is not already locked) for the duration of the <commit>.

Note: Datastores may be locked independently.

The <lock> operation fails in the following situations:

- NETCONF session or another entity (such as CLI) already has a lock in place for that datastore.
- The target configuration is <candidate/> and the shared candidate configuration has already been modified, and these changes have not been committed or discarded.
- The target configuration is <running/> and another NETCONF session has an ongoing confirmed commit.

Note: To ensure the most secure configuration workflow, Nokia recommends locking the candidate and running datastores prior to a configuration change.

Table 39: Parameters for the <lock> operation

Parameter	Description	Comments
<target>	Specifies the configuration datastore to lock. Available options are: • <candidate/> • <running/>	Mandatory No default value

Example: RPC request and reply messages for the <lock> operation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <lock>
        <target>
            <running/>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
        </target>
    </lock>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

11.4.12 unlock

The <unlock> operation releases a configuration lock previously obtained using the <lock> operation.

For the <unlock> operation to succeed:

- the specified lock must be active
- the <unlock> operation must be issued in the same session that issued the lock

Locks held by a session are released when that session is terminated.

Table 40: Parameters for the <unlock> operation

Parameter	Description	Comment
<target>	Specifies the configuration datastore to lock: • <candidate/> • <running/>	Mandatory

Example: RPC request and reply for an <unlock> operation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <unlock>
        <target>
            <running/>
        </target>
    </unlock>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

11.4.13 delete-config

The <delete-config> operation deletes the contents of a target configuration datastore. The running configuration datastore cannot be deleted.

Although the :url capability is supported by SR Linux, the <delete-config> operation on SR Linux does not support it.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Note: Use this operation with extreme caution; depending on the requirements, consider using the <discard-changes> operation instead.

Table 41: Parameter for the <delete-config> operation

Parameter	Description	Comment
<target>	Specifies the configuration datastore to delete. The only supported value is <startup/>.	Mandatory

A successful RPC results in an <rpc-reply> with an <ok/> XML node. An unsuccessful RPC results in an <rpc-error> containing the appropriate error information for the detected issue.

Example: RPC request and reply message for the <delete-config> operation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <delete-config>
        <target>
            <startup/>
        </target>
    </delete-config>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

11.4.14 close-session

The <close-session> operation closes the active session. The NETCONF server takes the following actions:

- releases any locks and resources associated with the session
- gracefully closes any associated connections
- rejects any additional NETCONF requests

Any currently accepted NETCONF requests, including the <close-session> operation, are completed before closing the active session.

Example: RPC request and reply for the <close-session> operation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <close-session/>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <ok/>
</rpc-reply>
]]>]]>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

11.4.15 validate

The <validate> operation checks the contents of a specified configuration against the underlying YANG data model. The operation checks for syntax-errors, missing parameters, references to undefined configuration data.

Note: A successful configuration validation using the <validate> operation does not gaurantee that a subsequent <commit> operation will succeed. Platform restrictions such as available memory are evaluated at the point of configuration commit.

Table 42: Parameters for the <validate> operation

Parameter	Description	Comment
<source>	Specifies the source configuration datastore or the URL of configuration file to validate. • <candidate/> • <startup/> • <running/> • <url> The path to a local or remote file. You must have the correct permissions to access the source file. • <config> The XML configuration, correct according to the YANG model and with correct namespacing, provided in anyxml format. For details about datastores, see Network Management Datastore Architecture (NMDA).	Mandatory No default value

Example: RPC request and response for the <validate> operation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <validate>
        <source>
            <candidate/>
        </source>
    </validate>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

11.4.16 action

The <action> element (YANG 1.1) invokes a YANG-modeled operation on SR Linux. The element is sent inside an <rpc> message. SR Linux does not define additional NETCONF RPC operations for YANG-modeled actions; you use the standard <action> encoding. The XML in the request and the structured data in a successful <rpc-reply> follow the input and output defined for that action in its YANG module.

Table 43: Content of the <action> element

Element	Description	Comment
Action and input	Input leaves defined in the model appear as child elements of that action element (not inside an <input> wrapper). The input provided when invoking a YANG action must match the structure and content specified in its corresponding YANG module definition. See the *SR Linux Data Model Reference Guide*.	Mandatory Structure is action-specific.

YANG-modeled operations over NETCONF are described in YANG-modeled operations - NETCONF and Invoking a YANG action.

11.4.17 kill-session

The <kill-session> operation terminates any NETCONF session, except the current session. To close the current session, use the <close-session> operation; see close-session for details. You cannot use the <kill-session> operation to terminate a session from another interface such as CLI or gRPC.

When a session is terminated, operations that are pending in that session are discarded and any locks held by that session are released. When a NETCONF server receives a <kill-session> request while processing a confirmed commit, it restores the running configuration to its state before the confirmed commit was issued.

Table 44: Parameters for the <kill-session> operation

Parameter	Description	Comment
<session-id>	Specifies the session ID of the NETCONF session to be terminated.	Mandatory

Example: RPC request and reply for the <kill-session> operation

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <kill-session>
        <session-id>146</session-id>
    </kill-session>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
]]>]]>
```

11.5 NETCONF options

This section outlines the SR Linux NETCONF implementation for the following options:

- `with-defaults`
- `with-oritin`
- options for the `operation` attribute

11.5.1 with-defaults option

The `with-defaults` option specifies how the NETCONF server handles the retrieval of default data. SR Linux NETCONF server supports the following behaviors for the `with-defaults` option:

- In `explicit` mode, the default behavior for SR Linux, the output contains all data that is not a system default. It returns all data where the value is the same as the system default but was explicitly set by the operator.
- In `report-all` mode, the output includes all data, including system defaults (whether explicit, implicit or system calculated).

The following SR Linux NETCONF operations support the `with-defaults` option:

- get
- get-config
- get-data
- copy-config (where the `target` is a URL)

The NETCONF server advertises the following capabilities:

- `urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=explicit&also-supported=report-all`
- `urn:ietf:params:netconf:capability:with-operational-defaults:1.0`

For the `with-defaults` option to function on the `operational` datastore, the `with-operational-defaults` capability is advertised by SR Linux along with the `with-defaults` capability. If either of these is missing and the operator attempts to use the `with-defaults` tag with the `operational` datastore, the RPC returns an `rpc-error` with the `invalid-value` `error-tag`.

Note: The `basic-mode` and `also-supported` parameters are attached to the `with-defaults` capability. These detail what the default behavior of the system is and what other behavior is available for use by an operator.

Example: RPC request and reply for the <get-config> operation using the <with-defaults> option in report-all mode

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
      <candidate/>
    </source>
    <with-defaults xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">report-
all</with-defaults>
    <filter>
      <system xmlns="urn:nokia.com:srlinux:general:system">
        <ssh-server xmlns="urn:nokia.com:srlinux:linux:ssh">
          <name>mgmt</name>
        </ssh-server>
      </system>
    </filter>
  </get-config>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <system xmlns="urn:nokia.com:srlinux:general:system">
      <ssh-server xmlns="urn:nokia.com:srlinux:linux:ssh">
        <name>mgmt</name>
        <admin-state>enable</admin-state>
        <network-instance>mgmt</network-instance>
        <port>22</port>
        <disable-shell>false</disable-shell>
        <timeout>0</timeout>
        <rate-limit>20</rate-limit>
        <host-key>
          <preserve>true</preserve>
        </host-key>
      </ssh-server>
    </system>
  </data>
</rpc-reply>
]]>]]>
```

11.5.2 `with-origin` option

The `with-origin` option details, for each configuration node (except non-presence containers which do not have an origin), where the data came from as described in RFC 8342. The `with-origin` option is only supported in conjunction with the operational datastore.

The following are the possible values for an elements origin:

- `origin`: Not used in SR Linux.
- `intended`: represents configuration provided by the intended configuration datastore. Most configuration elements are typically marked as `intended`.
- `dynamic`: represents configuration provided by a dynamic configuration datastore. No dynamic configuration datastores are provided in SR Linux.
- `system`: represents configuration provided by the system itself. Examples of system configuration include applied configuration for an always-existing loopback interface, or interface configuration that is auto-created because of the hardware currently present in the device.
- `learned`: represents configuration that has been learned via protocol interactions with other systems, including such protocols as link-layer negotiations, routing protocols, and DHCP.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

- `default`: represents configuration using a default value specified in the data model, using either values in the "default" statement or any values described in the "description" statement. The default origin is only used when the configuration has not been provided by any other source.
- `unknown`: represents configuration for which the system cannot identify the origin.

The following example shows the origin is provided in the `urn:ietf:params:xml:ns:yang:ietf-origin` namespace from the `ietf-origin` YANG module. Any origin is referenced using the correct namespace.

```
<rpc message-id="101" 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>
            <system xmlns="urn:nokia.com:srlinux:general:system">
                <control-plane-traffic/>
            </system>
        </subtree-filter>
    </get-data>
</rpc>
]]>]]>

<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-nmda">
        <system xmlns="urn:nokia.com:srlinux:general:system" xmlns:or=
"urn:ietf:params:xml:ns:yang:ietf-origin">
            <control-plane-traffic>
                <output>
                    <qos xmlns="urn:nokia.com:srlinux:qos:qos">
                        <management-protocols-dscp or:origin="or:intended">32</management-
protocols-dscp>
                    </qos>
                </output>
                <input>
                    <acl xmlns="urn:nokia.com:srlinux:acl:acl">
                        <acl-filter or:origin="or:intended">
                            <name>cpm</name>
                            <type>ipv4</type>
                        </acl-filter>
                        <acl-filter or:origin="or:intended">
                            <name>cpm</name>
                            <type>ipv6</type>
                        </acl-filter>
                    </acl>
                </input>
            </control-plane-traffic>
        </system>
    </data>
</rpc-reply>
]]>]]>
```

The `<origin>` tag reflects the source of the configuration that is in use by the system; this applies only to the configuration data (and only in the operational datastore), not state.

If the origin is ambiguous, such as when the same data node value has originated from multiple sources, SR Linux selects the most appropriate origin that node. If the node contains a protocol-negotiated value that does not override an explicitly configured value, the origin is reported as `intended`.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

11.5.3 Options for the operation attribute

The following options are available for the `operation` attribute:

Table 45: `operation` attribute options

Option	Description
`merge`	Merges the configuration data provided in this attribute with the configuration at the corresponding level in the configuration datastore identified by the `<target>` parameter.
`replace`	Replaces any related configuration in the configuration datastore identified by the `<target>` parameter with the configuration data identified by the element containing this attribute. If no such configuration data exists in the configuration datastore, it is created. Unlike the `<copy-config>` operation, which replaces the entire `target` configuration, only the configuration specified in the `<config>` or `<url>` parameters are affected.
`create`	Adds the configuration data identified by the element containing this attribute to the configuration only if the configuration data does not already exist in the configuration datastore. If the configuration data exists, an `<rpc-error>` element is returned with an `<error-tag>` value of `data-exists`.
`delete`	Deletes the configuration data identified by the element containing this attribute from the configuration, if the configuration data currently exists in the configuration datastore. If the configuration data does not exist, an `<rpc-error>` element is returned with an `<error-tag>` value of `data-missing`.
`remove`	Deletes from the configuration datastore configuration data identified by the element containing the remove attribute. If the configuration data to be removed does not exist, the `remove` operation is silently ignored by the server.

Related topics

edit-config

11.6 NETCONF filtering

NETCONF filtering is a mechanism that allows an application to select which YANG subtrees to include in the RPC reply message for those operations that support filtering. SR Linux supports subtree filtering.

XPath filtering is not supported on SR Linux.

11.6.1 Subtree filtering

Subtree filtering allows you to select which YANG subtrees (encoded in XML) to include in the `<rpc-reply>` message for operations that support filtering. Subtree filtering is described in RFC 6241.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

A subtree filter includes zero or more element subtrees, which represent the filter selection criteria. At each containment level within a subtree, the set of sibling nodes is logically processed by the server to determine if its subtree and the paths of elements to the root are included in the filter output. All elements present in a particular subtree within a filter must match associated nodes present in the server's YANG schema.

XML namespaces must be specified (via `xmlns` declarations) if required within the filter data model. The namespace must exactly match a namespace supported by the server.

Note: Prefix values for qualified namespaces are not relevant when comparing filter elements to elements in the underlying data model. Only data associated with a specified namespace is included in the filter output.

Each node specified in a subtree filter represents an inclusive filter. Only associated nodes in underlying data model within the specified configuration datastore on the server are selected by the filter. A node must match the namespace exactly and hierarchy of elements specified in the filter data, except that the filter absolute path name is adjusted to start from the layer below the `<filter>` context.

Response messages contain only the subtrees selected by the filter. Any selection criteria that were present in the request, within a particular selected subtree, are also included in the response. Some elements expressed in the filter as leaf nodes are expanded (that is, subtrees included) in the filter output. Specific data instances are not duplicated in the response if that request contains multiple filter subtree expressions that select the same data.

Note: Providing an empty filter will return no data. Providing no filter will return all data.

SR Linux supports the following filter components:

- Namespace selection and wildcarding
- Containment nodes
- Selection nodes
- Content match nodes

Note: Attribute match expressions are not supported on SR Linux.

11.6.1.1 Namespace selection and wildcarding

The filter output includes only elements from the specified namespace. A namespace is considered match if the content of the `xmlns` attribute is the same in the filter and the underlying data model. At least one element must be specified in the filter any elements to be included in the filter output.

If no namespace is provided to a filter element, the server evaluates for all namespaces that it is aware of.

Note: Typically, most elements are namespaced through inheritance, except for top-level nodes.

Consider the following data model:

```
module foo {
    namespace "urn:nokia.com:foo";
    container top {
        container my-container {
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
            leaf my-leaf { type string; }
        }
    }
}
```

```
module bar {
    namespace "urn.nokia.com:bar";
    container top {
        container my-container {
            leaf my-leaf { type string; }
        }
    }
}
```

Example: Filter with a specified namespace

```
<get>
    <filter>
        <top xmlns="urn:nokia.com:foo"/>
    </filter>
</get>
]]>]]>

<rpc-reply>
    <top xmlns="urn:nokia.com:foo">
        <my-container>
            <my-leaf>data from foo</my-leaf>
        </my-container>
    </top>
</rpc-reply>
]]>]]>
```

All data starting from the <top> node in the "urn:nokia.com:foo" namespace is returned. No data is returned from the "urn.nokia.com:bar" namespace as it has not been selected.

Example: Filter with multiple namespaces

```
<get>
    <filter>
        <top xmlns="urn:nokia.com:foo">
              <my-container xmlns="urn.nokia.com:bar"/>
        </top>
    </filter>
</get>
]]>]]>

<rpc-reply>
      <top xmlns="urn:nokia.com:foo">
      </top>
</rpc-reply>
]]>]]>
```

The data requested is from the <top> element in the "urn:nokia.com:foo" namespace and the <my-container> element in the "urn.nokia.com:bar" namespace. As there is no <my-container> element in the "urn.nokia.com:bar" namespace under the <top> container in the "urn:nokia.com:foo" namespace, no data is returned, however, the <top> container does exist in the requested "urn:nokia.com:foo" namespace so that is returned.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: Explicitly requesting an empty (wildcard) namespace

```
<get>
    <filter>
        <top xmlns=""/>
    </filter>
</get>
]]>]]>

<rpc-reply>
    <top xmlns="urn:nokia.com:foo">
        <my-container>
            <my-leaf>data from foo</my-leaf>
        </my-container>
    </top>
    <top xmlns="urn:nokia.com:bar">
        <my-container>
            <my-leaf>data from bar</my-leaf>
        </my-container>
    </top>
</rpc-reply>
]]>]]>
```

Example: Requesting the `network-instance` node defined in the SR Linux namespace

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <candidate/>
        </source>
        <filter>
            <network-instance xmlns="urn:nokia.com:srlinux:net-inst:network-instance"/>
        </filter>
    </get-config>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <network-instance xmlns="urn:nokia.com:srlinux:net-inst:network-instance"
 xmlns:nokia-attr="urn:nokia.com:sros:ns:yang:sr:attributes">
            <name>mgmt</name>
            <type>ip-vrf</type>
            <admin-state>enable</admin-state>
            <description>Management network instance</description>
            <interface>
                <name>mgmt0.0</name>
            </interface>
            <protocols>
                <linux xmlns="urn:nokia.com:srlinux:linux:linux">
                    <import-routes>true</import-routes>
                    <export-routes>true</export-routes>
                    <export-neighbors>true</export-neighbors>
                </linux>
            </protocols>
        </network-instance>
    </data>
</rpc-reply>
]]>]]>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Example: Requesting the `network-instance` node using a wildcard namespace

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <candidate/>
        </source>
        <filter>
            <network-instance xmlns=""/>
        </filter>
    </get-config>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <network-instance xmlns="urn:nokia.com:srlinux:net-inst:network-instance"
 xmlns:nokia-attr="urn:nokia.com:sros:ns:yang:sr:attributes">
            <name>mgmt</name>
            <type>ip-vrf</type>
            <admin-state>enable</admin-state>
            <description>Management network instance</description>
            <interface>
                <name>mgmt0.0</name>
            </interface>
            <protocols>
                <linux xmlns="urn:nokia.com:srlinux:linux:linux">
                    <import-routes>true</import-routes>
                    <export-routes>true</export-routes>
                    <export-neighbors>true</export-neighbors>
                </linux>
            </protocols>
        </network-instance>
    </data>
</rpc-reply>
]]>]]>
```

11.6.1.2 Containment nodes

A containment node in a subtree filter is a node that contains one or more child elements (that may themselves be containment nodes). In the following example, `system` and `logging` would be considered containment nodes.

Example: Filtering using containment nodes

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <candidate/>
        </source>
        <filter>
            <system xmlns="urn:nokia.com:srlinux:general:system">
                <logging xmlns="urn:nokia.com:srlinux:log:logging">
                    <file/>
                </logging>
            </system>
        </filter>
    </get-config>
</rpc>
]]>]]>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <system xmlns="urn:nokia.com:srlinux:general:system">
            <logging xmlns="urn:nokia.com:srlinux:log:logging">
                <file>
                    <file-name>messages</file-name>
                    <rotate>3</rotate>
                    <size>10000000</size>
                    <facility>
                        <facility-name>local6</facility-name>
                        <priority>
                            <match-above>warning</match-above>
                        </priority>
                    </facility>
                </file>
            </logging>
        </system>
    </data>
</rpc-reply>
]]>]]>
```

11.6.1.3 Selection nodes

An empty leaf node within a filter is called a selection node. It represents an explicit selection filter on the underlying data model. Presence of any selection nodes within a set of sibling nodes causes the filter to select the specified subtrees and suppresses automatic selection of the entire set of sibling nodes in the underlying data model. For filtering purposes, an empty leaf node can be declared either with an empty tag (for example, `<example/>` or with explicit start and end tags (for example, `<example> </example>`. Any white space character is ignored.

Example: Filtering using selection nodes

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <candidate/>
        </source>
        <filter>
            <system xmlns="urn:nokia.com:srlinux:general:system">
                <logging xmlns="urn:nokia.com:srlinux:log:logging">
                    <buffer>
                        <buffer-name/>
                    </buffer>
                </logging>
            </system>
        </filter>
    </get-config>
</rpc>
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <system xmlns="urn:nokia.com:srlinux:general:system">
            <logging xmlns="urn:nokia.com:srlinux:log:logging">
                <buffer>
                    <buffer-name>messages</buffer-name>
                </buffer>
                <buffer>
                    <buffer-name>system</buffer-name>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
                </buffer>
            </logging>
        </system>
    </data>
</rpc-reply>
]]>]]>
```

11.6.1.4 Content match nodes

A leaf node that contains content is called a content match node. It is used to select some or all of its sibling nodes for filter output, and it represents an exact-match filter on the leaf node element content. The following constraints apply to content match nodes:

- A content match node must not contain nested elements.
- Multiple content match nodes (that is, sibling nodes) are logically combined in an "AND" expression.
- Filtering of mixed content is not supported.
- Filtering of list content is not supported.
- Filtering of whitespace-only content is not supported.
- A content match node must contain non-whitespace characters. An empty element (for example, `<example> </example>`) is interpreted as a selection node (that is, `<example/>`).
- Leading and trailing whitespace characters are ignored, but any whitespace characters within a block of text characters are not ignored or modified.

If all specified sibling content match nodes in a subtree filter expression are true, the filter output nodes are selected in the following manner:

1. Each content match node in the sibling set is included in the filter output.
2. If any containment nodes are present in the sibling set, they are processed further and are included if any nested filter criteria are also met.
3. If any selection nodes are present in the sibling set, all of them are included in the filter output.
4. If there are no selection or containment nodes in the filter sibling set, all the nodes defined at this level in the underlying data model (and their subtrees, if any) are returned in the filter output. If any of the sibling content match node tests are 'false', no further filter processing is performed on that sibling set, and none of the sibling subtrees are selected by the filter, including the content match nodes.

Example: Filtering using content match nodes

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <candidate/>
        </source>
        <filter>
            <system xmlns="urn:nokia.com:srlinux:general:system">
                <logging xmlns="urn:nokia.com:srlinux:log:logging">
                    <buffer>
                        <buffer-name>messages</buffer-name>
                    </buffer>
                </logging>
            </system>
        </filter>
    </get-config>
</rpc>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
]]>]]>

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <system xmlns="urn:nokia.com:srlinux:general:system">
            <logging xmlns="urn:nokia.com:srlinux:log:logging">
                <buffer>
                    <buffer-name>messages</buffer-name>
                    <rotate>3</rotate>
                    <size>10000000</size>
                    <facility>
                        <facility-name>local6</facility-name>
                        <priority>
                            <match-above>informational</match-above>
                        </priority>
                    </facility>
                </buffer>
            </logging>
        </system>
    </data>
</rpc-reply>
]]>]]>
```

11.6.1.5 Subtree filter processing

After a filter output is applied, the filter output (the set of selected nodes) is initially empty. Each subtree filter can contain one or more data model fragments, which represent portions of the data model that should be selected (with all child nodes) in the filter output.

1. Each subtree data fragment is compared by the server to the internal data models supported by the server.
2. If the entire subtree data- fragment filter (starting from the root to the innermost element specified in the filter) exactly matches a corresponding portion of the supported data model, the node and all its children are included in the result data.
3. The server processes all nodes with the same parent node (sibling set) together, starting from the root to the leaf nodes. The root elements in the filter are considered in the same sibling set (assuming they are in the same namespace), even though they do not have a common parent.
4. For each sibling set, the server determines which nodes are included (or potentially included) in the filter output, and which sibling subtrees are excluded from the filter output. The server first determines which types of nodes are present in the sibling set and processes the nodes according to the rules for their type.
5. If any nodes in the sibling set are selected, the process is recursively applied to the sibling sets of each selected node. The algorithm continues until all sibling sets in all subtrees specified in the filter have been processed.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

12 SNMP

SR Linux supports the Simple Network Management Protocol (SNMP) versions SNMPv2c and SNMPv3, which allow SNMP managers to read information about the system for device monitoring.

12.1 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 from 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 actions can occur between the agent and the manager:

- The manager gets information from 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 protocol data units (PDUs) are supported over UDP. The following PDUs are supported over TCP:

- GetBulkRequest
- GetNextRequest
- GetRequest
- Response

12.2 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's management information 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 Internet Assigned Numbers Authority (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.

MIB files are packaged with each release and are available on the Nokia support portal or in `/opt/srlinux/snmp/MIBs.zip`.

A simple list of supported OIDs for monitoring is in `/etc/opt/srlinux/snmp/numbers.txt`, and a detailed list is in `/etc/opt/srlinux/snmp/exportedOids` when an `access-group` is configured. A

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

list of OIDs available for traps is in `/etc/opt/srlinux/snmp/installedTraps` when a `trap-group` is configured. These files are created at runtime when the SNMP agent is started.

SR Linux provides a customizable SNMP framework that allows you to define custom MIBs for get requests and traps. This framework also powers SR Linux's built-in MIBs and traps, providing flexibility to customize SNMP MIBs to the specific requirements for your network. The framework consists of:

- Mapping files (YAML): Define MIB tables and OIDs.
- Conversion scripts (uPython): Process data from the management server via gNMI and convert it for SNMP.

See the "SNMP" chapter of the *SR Linux System Management Guide* for more information.

12.3 SNMP network instance configuration

About this task

The SNMP agent must first be configured to run in each network instance used to monitor the system. Then, access groups can be configured to read information or trap groups can be configured to send traps.

To configure the SNMP agent, use the **system snmp** commands.

Example: SNMP network instance configuration

In the following example, the SNMP agent is running in the **default** network instance.

```
--{ * candidate shared default }--[  ]--
# info with-context system snmp
    system {
        snmp {
            network-instance default {
                admin-state enable
            }
        }
    }
```

12.4 SNMP versions and configuration

The SNMP agent supports two versions of the SNMP protocol:

- 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.

12.4.1 SNMPv3 authentication and privacy protocols

The User-based Security Model (USM) for the authentication, encryption, and decryption of SNMPv3 packets is supported with configurable authentication and privacy protocols.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

SNMPv3 authentication 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

SNMPv3 privacy protocols

The following SNMPv3 privacy protocols are supported:

- CBC-DES
- CFB128-AES-128
- CFB128-AES-192
- CFB128-AES-256

SNMPv3 authentication and privacy protocol combinations

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)

12.4.2 Configuring SNMPv2c

About this task

SR Linux supports SNMPv2c, which allows SNMP managers to read information about the system for device monitoring.

Procedure

To configure the SNMP agent, use the **system snmp** commands.

Example: SNMPv2c access group configuration

In SNMPv2c, the **community** value is mandatory and cannot contain spaces.

Optionally, the **prefix-list** value defines which managers can use the community (both IPv4 and IPv6 addresses) and is only supported in SNMPv2c.

The **community-entry** value cannot be the same as the **community** value because this reveals the plaintext value of the **community**.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

In the following example, the SNMPv2c agent uses an access group for get requests. The minimum security level is configured.

```
--{ * candidate shared default }--[  ]--
# info with-context system snmp
    system {
        snmp {
            access-group ag1 {
                admin-state enable
                security-level no-auth-no-priv
                community-entry ce1 {
                    community $aes1$AW/5wLmAOcTPhG8=$aFJfMhdHwSGTplCfsDgBPA==
                    prefix-list [
                        10.1.1.1/32
                    ]
                }
            }
            network-instance default {
                admin-state enable
            }
        }
    }
```

Example: SNMPv2c trap group configuration

In the following example, the SNMPv2c agent uses a trap group within the default network instance. The minimum security level is configured. The SNMPv2c **community** value is configured using **community-entry**.

```
--{ * candidate shared default }--[  ]--
# info with-context system snmp
    system {
        snmp {
            trap-group tg1 {
                admin-state enable
                network-instance default
                destination destination1 {
                    admin-state enable
                    address 10.2.2.2
                    security-level no-auth-no-priv
                    community-entry ce1 {
                        community $aes1$AWOTWOQo41n22m8=$XD4pX1F7pWJFtTdgwjf23w==
                    }
                }
            }
            network-instance default {
                admin-state enable
            }
        }
    }
```

12.4.3 Configuring SNMPv3

About this task

SR Linux supports SNMPv3, which allows SNMP managers to read information about the system for device monitoring.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Procedure

To configure the SNMP agent, use the **system snmp** commands.

Example: SNMPv3 access group configuration

In the following example, the SNMPv3 agent uses an access group for get requests. The SNMPv3 user authentication and privacy protocols are configured using **security-entry**. The value of **password** cannot contain spaces and the user names in access groups and trap groups must be unique.

```
--{ * candidate shared default }--[  ]--
# info with-context system snmp
    system {
        snmp {
            access-group ag1 {
                admin-state enable
                security-level auth-priv
                security-entry se1 {
                    user getuser1
                    authentication {
                        protocol hmac-md5-96
                        password $aes1$AW8qEdNV+4KmIm8=$F2zgIDAO4DkcFh+6oLyd2w==
                    }
                    privacy {
                        protocol cbc-des
                        password $aes1$AW+ZudVoGPQP5W8=$1UMEKehkoPqo8zGFE0KzxA==
                    }
                }
            }
            network-instance default {
                admin-state enable
            }
        }
    }
```

Example: SNMPv3 trap group configuration

In the following example, the SNMPv3 agent uses a trap group within the default network instance. The SNMPv3 user authentication and privacy protocols are configured using **security-entry**. The value of **password** cannot contain spaces and the user names in access groups and trap groups must be unique.

```
--{ * candidate shared default }--[  ]--
# info with-context system snmp
    system {
        snmp {
            trap-group tg1 {
                admin-state enable
                network-instance default
                destination destination1 {
                    admin-state enable
                    address 10.2.2.2
                    security-level auth-priv
                    security-entry se1 {
                        user trapuser1
                        authentication {
                            protocol hmac-md5-96
                            password $aes1$AW/ZYq/e/AbNS28=$Dw3ipXdBawX9P10lfe/zAw==
                        }
                        privacy {
                            protocol cbc-des
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
                        password $aes1$AW9Bvh9EbJcvwm8=$V214DJcpHCvdxkWbewDVuQ==
                    }
                }
            }
        }
        network-instance default {
            admin-state enable
        }
    }
}
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

13 Configuration secrets

SR Linux supports protection for the following basic types of configuration secrets:

- passwords
- shared secrets
- primary secret

13.1 Passwords

Passwords in SR Linux are protected using one-way hashing. Values stored using one-way hashing cannot be reversed; the plaintext form cannot be retrieved by the system. The hashed value is stored in the configuration.

SR Linux can verify a user password by comparing hashed password values. The system hashes the configured plaintext password and stores the hashed value in the configuration. During authentication, the system hashes the password the user enters with the same algorithm, then compares the result to the hashed value stored in the configuration. If they match, authentication succeeds.

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

Password hashing is applied to all locally configured users, including the admin and the linuxadmin user. SR Linux supports `yescrypt` encryption in addition to the argon2 (`ar2`) and SHA512 (`sha2`) algorithms.

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

Use the **system aaa authentication password hash-method** command to configure the algorithm used to hash user passwords.

Example

The following example changes the hash-method for user passwords to the argon2 (`ar2`) algorithm.

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


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

13.2 Shared secrets

Shared secrets are values that the router needs to recover in plaintext form to perform various protocol functions, such as peer authentication in routing protocols or RADIUS passwords. Shared secrets are protected in SR Linux using reversible encryption. The encrypted value is displayed and stored in the configuration.

SR Linux supports several encrypted formats for shared secrets. The first few characters (prefix) of the encrypted value indicate the format (for example, `$aes1$` or `$cryp3$`). The following is an example.

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp authentication
    network-instance default {
        protocols {
            bgp {
                authentication {
                    password $aes1$ATQyHDi8sc1ad28=$62q1s7X2WBBsCo2+Hf+McXWBVsL1VKRz7LipfWAe
Mdo=3
                }
            }
        }
    }
```

Both the `$aes1$` and `$cryp3$` formats use AES encryption.

The encryption key for `$cryp3$` is derived from a user-created primary secret using PBKDF2. If a primary secret exists on the router, the system uses `$cryp3$` for shared secrets. See Primary secret for more information.

A configuration file that contains `$cryp3$` secrets cannot be copied from one router to another when each router has a unique primary secret. For chassis replacement, the same primary secret can be created on the new router, and the configuration file can then be copied over from the old router. Subsequently, the primary secret on the new router can be changed if needed.

13.3 Primary secret

SR Linux supports a user-configurable primary secret that is used to derive an encryption key for securing reversible shared secrets in the configuration.

You configure a primary secret using a **tools** command, as shown in the following example:

```
--{ +* candidate shared default }--[  ]--
# tools system storage-encryption primary-secret set-secret
Enter new-secret: <12-128 byte string>
Re-enter new-secret: <12-128 byte string>
```

When a primary secret is configured on a router, the `$cryp3$` encryption format is used for securing shared secrets in the configuration. After a primary secret is created, the configuration file cannot be copied from one router to another. See Shared secrets for more information.

Setting or changing the primary secret causes the main configuration to be saved to disk. The primary secret itself is not stored in the configuration file, but it is persistent across reboots.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

When first adding a primary secret or changing an existing primary secret, previous copies of the configuration (including checkpoints) remain on disk and have shared secrets protected with the method that was previously in use. If needed, those files must be manually deleted by an administrator.

Note: Nokia recommends creating a unique primary secret on each router generated from a high-entropy source.

Note: The primary secret cannot be seen by any user, including administrators or Nokia support. It should be securely stored offline in password-protected storage for emergency recovery purposes.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

14 Modular Database (MDB)

Note: MDB profiles are supported only on 7250 IXR Gen 3 and 7250 IXR Gen 2c+ platforms.

Datapath tables used to store forwarding, routing and related entries, such as ARP/ND tables, IP FIB, can be re-partitioned to support different scale profiles. These tables are all governed by the Modular Database (MDB). MDB determines how datapath resources are allocated across different lookup tables. The datapath lookup tables are essential for packet forwarding, such as IP routing, MAC addresses, MPLS NHLFEs, and VPN labels, among others. With a different MDB profile, resources can be reallocated across lookup tables, allowing a specific table to accommodate more entries based on network deployment use cases.

For example, in deployments that require a large IP FIB, the resources allocated to other use-cases such as MPLS, MAC-VRF Forwarding Database (FDB) can be trimmed down or even zeroed out to support a higher IP FIB scale.

7250 IXR platforms support different MDB profiles. During system boot-up, based on the MDB profile configuration, the router datapath resources are allocated for specific deployment needs.

MDB profile update requires a system reboot for changes to take effect.

The SR Linux routers support both default and custom profiles optimized to meet specific network requirements. For details on supported profiles, see MDB profiles.

14.1 MDB profiles

The following MDB profiles are supported:

Table 46: Profile ID allocation for 7250 IXR Gen 2c+ platforms

Profile ID	Description
1	General-purpose default profile suitable for most use cases, including support for BGP-LU and SR-MPLS applications
2	Provides support to manage fallback routing, achieved through a secondary FIB lookup in the default network instance (`system.datapath.secondary-default-lookup`)
3	Distinguished by its support for IP multicast and supports a higher number of MPLS ILM entries, suitable for BGP Labeled Unicast (BGP-LU) and SR-MPLS applications.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Table 47: Profile ID allocation for 7250 IXR Gen 3 platforms

Profile ID	Description
1	General-purpose default profile suitable for most use cases.
4	Optimized for MPLS and labeled prefix use cases while maintaining a large FIB.

14.2 Configuring an MDB profile

Procedure

To configure MDB profiles, set the parameter **id** value within the context **platform.resource-management.mdb-profile** to an applicable profile ID and reboot the node for the new profile to be active.

Note: The **chassis-reboot-required** within the context **platform.resource-management.mdb-profile** is automatically set to `true` when a new MDB profile ID is configured.

Example: Display the active MDB profile ID

The following configuration examples displays the configured active MDB profile ID.

```
--{ + running }--[  ]--
# info with-context platform resource-management mdb-profile
    platform {
        resource-management {
            mdb-profile {
                id 1
            }
        }
    }
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

15 General and operational commands

The following table defines general and operational commands that can be entered at any context in the CLI.

Table 48: General and operational commands

Command	Description
/	Moves to the root; can also be used to reset the context to the root for a specific command (for example: **info from state /**)
?	Displays the context-based help
back	Returns to the previous context where the last command was executed
baseline <*argument*>	Arguments: • check - check baseline update for current candidates • diff - show baseline configuration changes • update - update baseline for current candidate
bash	Opens a Linux Bash session
bash <*command*>	Executes a Linux Bash command without entering session
cls	Clears the screen
diff [flat]	Compares the current candidate against the running configuration. Specifying **flat** provides a copy/paste format showing inserts and deletes. Global arguments include: • **baseline** - configuration candidate baseline • **candidate** - configuration candidate • **checkpoint** - configuration checkpoint • **factory** - factory configuration • **file** - file with configuration stored in .json format • **from** - source datastore to compare with • **rescue** - rescue configuration • **running** - running datastore • **startup** - startup configuration If arguments are not used, this command must be used in candidate mode.
echo	Echoes text back to a session

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Command	Description
enter	Switches to a different mode
enter candidate	Enters the shared candidate mode. Shared candidate mode is the default.
enter candidate exclusive	Enters the exclusive candidate mode To switch between shared and exclusive, you must switch to a different datastore (for example. running).
enter candidate exclusive name <*name*>	Enters the exclusive mode for a named candidate
enter candidate name <*name*>	Enters the shared candidate mode for a named shared candidate
enter candidate private	Enters the candidate private mode
enter candidate private name <*name*>	Enters the candidate private mode for a named candidate
enter running	Enters the running mode (default)
enter show	Enters the show mode (used with show CLI plug-ins)
enter state	Enters the state mode (all configuration and operational states)
environment alias *name*	Creates or overwrites an alias
environment bottom-toolbar *format*	Sets the text displayed in the bottom toolbar
environment cli-engine completion-ignore-case *value*	Ignores the word case in command auto-completion
environment cli-engine completion-type <*value*>	Sets the command auto-completion type
environment cli-engine history-filename <*value*>	Sets the command history file name
environment cli-engine max-history-items <*value*>	Sets the number of items in the command history
environment cli-engine refresh-interval <*value:0*>	Sets the bottom toolbar refresh interval
environment cli-engine type <*value*>	Sets the CLI engine type
environment cli-engine vi-editing-mode <*value*>	Sets VI editing mode

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Command	Description
environment complete-on-space <*value*>	Sets command auto-completion with **Enter** and **Spacebar**, in addition to **Tab**
environment delete	Resets the environment to the system default settings
environment key-completer-limit <*limit*>	Sets the number of keys displayed in command auto-completion
environment load	Loads the environment settings from a file • **file** <*file*> • **home** (loads `~/.srlinuxrc`)
environment network-instance <*value*>	Sets the default network instance for administrative commands such as ping and traceroute
environment output-format <*value*>	Sets the CLI output format
environment pagination <*value*>	Sets the pagination behavior
environment prompt <*format*>	Sets the text displayed in the prompt before every input line
environment save	Saves the current environment to a file • **file** <*file*> • **home** (saves `~/.srlinuxrc`
environment session-idle-timeout <*value*>	Sets the session idle timeout
environment show	Shows the current environment settings
environment yang-models <*value*>	Sets the default data model used in the CLI
exit	Exits to the previous context
exit to <*ancestor*>	Exits to a specific ancestor of the current context
exit all	Exits to the root
filter	Filters output for show and info commands. Usage: `filter [<node>] [depth <value>] [fields <value>] [keys-only] [non-zero]` • **node** - Positional node to filter on • **depth** - Filter out sub-nodes that are more than *n* levels deeper • **fields** - Fields to include

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Command	Description
	• **keys-only** - Hide all fields • **non-zero** - Show only non-zero values (numeric values only)
file cat	The **file** commands allow you to interact with files and directories on the system. These commands are passed through to the Linux binaries that perform file actions, for example **cat** and **cp**. The **file** commands are supported in candidate, running, and state modes. **file cat** displays files contents Usage: **cat** *<path (1-255 times)>* **[-v] [-T] [-t] [-s] [-n] [-E] [-e] [-A] [--version]** • *path* - Specifies the path to a file or directory • **-v** - Verbose, explains what is being done • **-T** - Shows tabs, displaying TAB characters as `^I` • **-t** - Equivalent to **-vT** • **-s** - Suppresses repeated empty output lines • **-n** - Numbers all output lines • **-E** - Displays $ at the end of each line • **-e** - Equivalent to **-vE** • **-A** - Shows all, equivalent to **-vET** • **--version** - Outputs version information and exits
file cd	Changes working directory Usage: **cd [**<*path*>**]** • *path* - Specifies the path to a directory
file cp	Copies files and directories Usage: **cp** *<path (2 times)>* **[-a] [-f] [-i] [-l] [-n] [-r] [-p] [-R] [-s] [-u] [--version] [-v]** • *path* - Specifies the path to a file or directory • **-a** - Specifies archive mode, preserving file ownership, permissions, and timestamps • **-f** - If an existing destination file cannot be opened, removes it and tries again (this option is ignored when the **-n** option is also used) • **-i** - Interactively prompts before overwrite (overrides a previous **-n** option) • **-l** - Hard links files instead of copying • **-n** - Does not overwrite an existing file (overrides a previous **-i** option)

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Command	Description
	• **-r** - Copies directories recursively • **-p** - Preserves mode, ownership, and timestamps • **-R** - Copies directories recursively (alias of **-r**) • **-s** - Makes symbolic links instead of copying • **-u** - Copies only when the source file is newer than the destination or when the destination is missing • **--version** - Outputs version information and exits • **-v** - Verbose, explains what is being done
file ls	Lists directory contents Usage: **ls [**<*path (0-255 times)*>**] [-a] [-A] [-C] [--color <always|auto|never>] [-d] [-F] [-h] [-i] [-l] [-r] [-R] [-s] [-S] [--time-style <long-iso|full-iso|iso|locale>] [-t] [-u] [-U] [-v] [-x] [-X] [-1] [--version]** • *path* - Specifies the path to a file or directory • **-a** - Lists all, including entries starting with . • **-A** - List almost all, excluding implied . and .. • **-C** - Lists entries by columns • **--color** - Colorizes the output • **-d** - Lists directories themselves, not their contents • **-F** - Classifies, appending indicators (one of */=>@|) to entries • **-h** - Human readable, with **-l** or **-s**, prints human readable sizes (for example, 1K, 234M, 2G) • **-i** - Inode, prints the index number of each file • **-l** - Uses a long listing format • **-r** - Reverses order while sorting • **-R** - Lists subdirectories recursively • **-s** - Prints the allocated size of each file, in blocks • **-S** - Sorts by file size, largest first • **--time-style** - With **-l**, shows times using a specific style • **-t** - Sorts by modification time, newest first • **-u** - With **-lt**, sorts by, and shows access time; with **-l** shows access time and sorts by name; otherwise, sorts by access time, newest first • **-U** - Does not sort; lists entries in directory order • **-v** - Verbose, explains what is being done • **-x** - Lists entries by lines instead of by columns

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Command	Description
	• **-X** - Sorts alphabetically by entry extension • **-1** - Lists one file per line • **--version** - Outputs version information and exits
file md5sum	Calculates and verifies checksums. Usage: **md5sum [**<*path (0-255 times)*>**] [-c] [--ignore-missing] [--quiet] [--status] [--version]** • **-c** - Reads checksums from the files and checks them • **--ignore-missing** - Does not fail or report the status for missing files • **--quiet** - Does not print OK for each successfully verified file • **--status** - Does not output anything, status code shows success • **--version** - Outputs version information and exits
file mkdir	Creates directories. Usage: **mkdir** <*path (1-255 times)*> **[-m** <*value*>**] [-p] [--version] [-v]** • *path* - Specifies the path to a file or directory • **-m** - Sets the file mode (as in chmod), not a=rwx - umask • **-p** - No error if existing, makes parent directories as needed • **--version** - Outputs version information and exits • **-v** - Verbose, explains what is being done
file mv	Moves or renames files and directories Usage: **mv** <*path (2 times)*> **[-f] [-i] [-n] [-u] [--version] [-v]** • *path* - Specifies the path to a file or directory • **-f** - Force, does not prompt before overwriting • **-i** - Interactive, prompts before overwrite • **-n** - No clobber, does not overwrite an existing file if you specify more than one of **-i**, **-f**, **-n**, only the final one takes effect • **-u** - Update, moves only when the source is newer than the destination or when the destination is missing • **--version** - Outputs version information and exits • **-v** - Verbose, explains what is being done
file pwd	Prints the working directory Usage: **pwd [**<*path*>**] [--version]** • *path* - Specifies the path to a file or directory

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Command	Description
	• **--version** - Outputs version information and exits
file rm	Removes files or directories Usage: **rm** <*path (1-255 times)*> **[-f] [-i] [-I] [-r] [-R] [-d] [--version] [-v]** • *path* - Specifies the path to a file or directory • **-f** - Force, ignores nonexistent files and arguments, never prompts before removal • **-i** - Interactive, prompts before every removal • **-I** - Interactive, prompts once before removing more than three files, or when removing recursively; less intrusive than **-i**, while still giving protection against most mistakes • **-r** - Removes directories and their contents recursively • **-R** - Removes directories and their contents recursively (alias of **-r**) • **-d** - Directories, removes empty directories • **--version** - Outputs version information and exits • **-v** - Verbose, explains what is being done
file touch	Changes file timestamps Usage: **touch** <*path (1-255 times)*> **[-a] [-c] [-d** <*value*>**] [-m] [-r** <*value*>**] [-t** <*value*> **] [--version]** **-a** - Changes only the access time **-c** - Does not create any files **-d** - Parses this string and uses it instead of the current time **-m** - Changes only the modification time **-r** - Uses this file's times instead of the current time **-t** - Uses [[CC]YY]MMDDhhmm[.ss] instead of the current time **--version** - Outputs version information and exit
help	Displays mode-related help
history	Displays the command history list with line numbers
history hot	Displays the top 5 most frequently used commands
history clear	Clears the history
info [*path***]**	Shows the value of all nodes and fields under the current context; optionally made more specific by a path
info depth <*n*>	Filters out sub-nodes that are deeper than the specified depth

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Command	Description
info detail	Shows also default values for unset fields
info flat	Shows each node or field as a single line
info use-proto-json	Shows the output as the JSON used for the protobuf messages
info from <*mode*>	Executes the info command from within the specified mode (either candidate, running, or state). The current context can be retrieved without a from argument.
info with-context	Displays the full context of the configuration, whether the command is entered from the root context or from within a specified context.
list	Show the keys of all nodes under the current context
monitor [*path***]**	Monitors state changes within the current context; optionally made more specific by a path
monitor recursive [*path***]**	Includes children when monitoring
monitor sample	Uses sampling instead of on-change monitoring (interval in seconds)
ping	Sends IPv4 ICMPv4 echo requests to network hosts. Usage: **ping** <*destination*> **[-I** <*value*>**] [-M** <*value*>**] [-Q** <*value*>**] [-c** <*value*>**] [-i** <*value*>**] [-s** <*value*>**] [-t** <*value*>**] [network-instance** <*value*>**]** • **-I** - Source interface/IP • **-M** - Path MTU discovery strategy • **-M- pmtudisc_options** - sets df-bit set • **-Q** - tos • **-c** - Number of ping requests • **-i** - Wait interval in seconds between each packet • **-s** - Packet size • **-t** - TTL (Time-To-Live)
ping6	Sends IPv6 ICMPv6 echo requests to network hosts. Usage: **ping6** <*destination*> **[-I** <*value*>**] [-M** <*value*>**] [-Q** <*value*>**] [-c** <*value*>**] [-i** <*value*>**] [-s** <*value*>**] [-t** <*value*>**] [network-instance** <*value*>**]** • **-I** - Source interface/IP • **-M** - Path MTU discovery strategy • **-M- pmtudisc_options** - sets df-bit set • **-Q** - tos

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Command	Description
	• **-c** - Number of ping requests • **-i** - Wait interval seconds between each packet • **-s** - Packet size • **-t** - TTL (Time-To-Live)
pwc	Prints the current working context
quit	Closes the CLI session
save	Saves the current datastore to the specified file in JSON format. Usage: **save [detail] file** <*value*> **[from <running|state>] [text]** • **detail** - Additionally saves the default values for unset fields • **file** - Output filename • **from** - Datastore used to retrieve data from • **text** - Use CLI (text) format instead of JSON
show	Displays plug-in style show commands; see Pre-defined show reports.
source <*file*>	Executes a set of commands from a file
tech-support	Generates a technical support file. Usage: **tech-support [ignore-host-keys <***value***>] [max-time <***value***>] [network-instance <***value***>] [no-core] [scp-to <***value***>]** • **ignore-host-keys** - Skip security verification of remote SSH server key • **max-time** - Maximum time • **network-instance** - Network-instance for scp command • **no-core** - Skip inclusion of core files • **scp-to** - scp location, for example, user@server-ip:/tmp/
tools	Executes a tool command
traceroute	Prints the route packets trace to network host Usage: **traceroute** <*destination*>
traceroute6	Prints the route IPv6 packets trace to network host Usage: **traceroute6** <*destination*>
tcptraceroute	tcptraceroute compatible wrapper for traceroute Usage: **tcptraceroute** <*destination*>

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Command	Description
tree [*path***]**	Shows the tree structure in the current context; optionally made more specific with a path
tree flat	Shows each structure on a single line
tree from [*mode***]**	Retrieves the tree from the current context in another mode
watch	Execute a program periodically
Within candidate mode only, the following apply	
!!	Appends a line to the annotation of the current node
!!!	Replaces the annotation of the current node
commit now	Applies the changes, exits candidate mode, and enters running mode
commit stay	Applies the changes and then remains in candidate mode. Permitted additional arguments: commit stay [save] [comment] [confirmed]
commit save	Applies the changes and then remains in candidate mode. Permitted additional arguments: commit [stay] [checkpoint] save [confirmed] [comment]
commit checkpoint	Causes an automatic checkpoint after the commit succeeds. Permitted additional arguments: commit [stay] [now] checkpoint [save] [confirmed]
commit validate	Verifies that a propose configuration change passes a management server validation
commit comment <*comment*>	Used with other arguments (except **validate**) to add a user comment where comment is a quoted string, 1-255 characters. Permitted additional arguments: commit [stay] [save] [checkpoint] [confirmed] comment
commit confirmed **commit confirmed [timeout** <*1-86400*>**]** **commit confirmed [accept| reject]**	Applies the changes, but requires an explicit confirmation to become permanent. The timeout period default is 600 seconds (10 mins.), or can be provisioned with a value of 1-86400 sec.). The timeout argument cannot be used with the accept or reject parameter. Before the timer expires, the accept argument explicitly confirms and applies the changes. With no timer running, the reject argument explicitly rejects the changes.
annotate	Sets the annotation of the current node

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Command	Description
discard [now|stay]	Discards all uncommitted changes; requires either a **now** or **stay** option (to stop unintended commit). When **stay** is used, the mode remains in candidate mode (opening a new transaction).

insert	Inserts the provided value into the specified user-ordered list.
	Usage: **insert <***path-to-list***> <***value***> <first|before|after|last><** *value***>**
	The provided value can be a single entry or a list. Use square brackets when specifying a list. In these examples, the provided values get added to an example `community-set` list named *sample*.
	`insert community-set sample member [ 1:1 1:2 ]`
	`insert community-set sample member 1:1`
	• **first** - Insert a value at the beginning of the user-ordered list
	• **before** - Insert a value in the user-ordered list before another specified value
	• **after** - Insert a value in the user-ordered list after another specified value
	• **last** - Insert a value at the end of the user-ordered list
	The optional **after** or **before** parameters, or the **first** and the **last** parameters, are mutually exclusive to place the value or values at an exact location in the list.
	Note: These parameters cannot be used for a system-ordered list.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

16 Pre-defined show reports

The SR Linux CLI is a python application that can load dynamic libraries from other applications. This flexibility allows users to create their own show commands, and also allows each application to own its own show command tree. In addition to custom show commands, Nokia provides pre-defined plug-ins.

The following sections provide the commands and syntax for pre-defined python plug-ins.

Users can also create their own customer reports using CLI Plug-ins. For more information, see the *SR Linux CLI Plug-In Guide*.

16.1 ACL show reports

```
show
    - acl
        - acl-filter name type value
            - entry sequence-id [subinterface subinterface name
        - gbp-filter network-instance
            - entry sequence-id
        - summary
```

16.1.1 ACL descriptions

Context	Description
show acl	Container for ACL show reports.
show acl acl-filter *name* **type** *value*	Show a list of ACL filter policies. Specify the ACL filter name and type.
show acl acl-filter *name* **type** *value* **entry** *sequence-id* **[subinterface** *subinterface name***]**	Show ACL filter policies by entry ID.
show acl gbp-filter *network-instance*	Show GBP filter policies.
show acl gbp-filter *network-instance* **entry** *sequence-id*	Show GBP filter policies by entry ID.
show acl summary	Show a summarized list of all ACL filters.

16.2 ARP/ND show reports

```
show
    - arpnd
        - neighbors
            - interface <interface name>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
        – arp-entries
            – interface <interface name>
```

16.2.1 ARP/ND descriptions

Context	Description
show arpnd	Container for ARPND show reports.
show arpnd neighbors	Show all ARPND neighbors including associated interfaces, subinterfaces, origin, and state.
show arpnd neighbors interface *interface name*	Show ARPND neighbor for a specified interface name.
show arpnd arp-entries	Show all ARPND entries including associated interfaces, subinterfaces, origin, and expiry.
show arpnd arp-entries interface *interface name*	Show ARPND entries for a specified interface name.

16.3 Interface show reports

```
show
    – show interface <interface port name>
        – all
        – brief
        – detail
        – queue-detail
```

16.3.1 Interface descriptions

Context	Description
show interface *interface port name*	Container for Interface show reports. Can be used to show a report of all interfaces and subinterfaces with an operational state of up, or for a specified interface if an optional name is entered. Interfaces that are operationally down are not displayed; use the **all** option to display both up and down interfaces.
show interface *interface port name* **all**	Show a report that displays all up and down interfaces and subinterfaces, or for a specified interface if an optional name is entered. For interfaces with an operational state of up, IP addresses are displayed in addition to speed and type. For interfaces with an operational state of down, a reason for this state is provided.
show interface *interface port name* **brief**	Show a report that provides a summarized view of all interfaces and subinterfaces, or for a specified interface if an optional name

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
	is entered. Displays the administrative state, operational state, speed, and type.
show interface *interface port name* **detail**	Show a detail report for all up and down interfaces and subinterfaces, or for a specified interface if an optional name is entered. In addition to operational state, last changed, flow control, and MAC address details, this report includes queue parameters, statistics, and transceiver/transceiver channel details.
show interface *interface port name* **queue-detail**	Show an egress queue and VOQs report for all interfaces and subinterfaces, or for a specified interface if an optional name is entered.

16.4 IPv4 tunnel show reports

```
show
    – [network-instance name] ipv4 tunnel
        – detail
        – summary
```

16.4.1 IPv4 tunnel descriptions

Context	Description
show [network-instance *name* **] ipv4 tunnel [** *ipv4-prefix* **] [type** *value* **] [internal-tag** *value* **] [next-hop** *value* **] [or-longer]**	Show full IPv4 tunnel table report. If no network instance is specified the default network instance is assumed.
show [network-instance *name* **] ipv4 tunnel [** *ipv4-prefix* **] [type** *value* **] [internal-tag** *value* **] [next-hop** *value* **] [or-longer] detail**	Show detailed IPv4 tunnel table report. If no network instance is specified the default network instance is assumed.
show [network-instance *name***] ipv4 tunnel summary**	Show summary IPv4 tunnel table report. If no network instance is specified the default network instance is assumed.

16.5 IPv6 tunnel show reports

```
show
    – [network-instance name] ipv6 tunnel
        – detail
        – summary
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

16.5.1 IPv6 tunnel descriptions

Context	Description
show [network-instance *name* **] ipv6 tunnel [** *ipv6-prefix* **] [type** *value* **] [internal-tag** *value* **] [next-hop** *value* **] [or-longer]**	Show full IPv6 tunnel table report. If no network instance is specified the default network instance is assumed.
show [network-instance *name***]] ipv6 tunnel [** *ipv6-prefix* **] [type** *value* **] [internal-tag** *value* **] [next-hop** *value* **] [or-longer] detail**	Show detailed IPv6 tunnel table report. If no network instance is specified the default network instance is assumed.
show [network-instance *name***] ipv6 tunnel summary**	Show summary IPv6 tunnel table report. If no network instance is specified the default network instance is assumed.

16.6 LAG show reports

```
show
    – show lag <lag instance>
        – brief
        – detail
        – lacp-state
        – lacp-statistics
        – member-statistics
        – queue-detail
```

16.6.1 LAG descriptions

Context	Description
show lag *lag instance*	Container for lag show reports. Can be used to show a report of all LAGs, or for a specified LAG instance if an optional ID is entered.This includes a summary report of all operationally up and down LAGs.
show lag *lag instance* **brief**	Show a report that provides a summarized view of all LAGs, or for a specified LAG if an optional ID is entered. Displays the administrative state, operational state, aggregate speed, and min and active links.
show lag *lag instance* **detail**	Show a detailed report for all LAGs, or for a specified LAG if an optional ID is entered. This report includes the operational status of the LAG, list of member links and their operational status, aggregated queue statistics, aggregated traffic/error statistics, and aggregated traffic rate.
show lag *lag instance* **lacp-state**	Show a LAG report that details all LAGs or a specific LAG if it is enabled. Report also includes the LACP configuration mode and

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
	per member link statistics state details (operation state, activity, timeout, and so on).
show lag *lag instance* **lacp-statistics**	Show a LAG report that details all LAGs or specific LAG statistics. Report can include per member link statistics such as LACP in-pkts, out-pkts, rx-errors, tx-errors, unknown errors, and LACP errors.
show lag *lag instance* **member-statistics**	Show a LAG report that displays all member-statistics.
show lag *lag instance* **queue-detail**	Show a LAG report that displays aggregated queue statistics for all LAG link members.

16.7 MPLS show reports

```
show
    – [network-instance name] mpls
        – fib [label]
        – fib [label] detail
        – fib summary
```

16.7.1 MPLS descriptions

Context	Description
show [network-instance *name* **] mpls**	Container for MPLS reports.
show [network-instance *name* **] mpls fib [***label* **]**	Show full MPLS FIB report
show [network-instance *name* **] mpls fib [***label* **] detail**	Show detailed MPLS FIB report
show [network-instance *name* **] mpls fib summary**	Show summary MPLS FIB report

16.8 Network-instance show reports

```
show
    – network-instance <name>
        – bridge-table
            – mac-duplication duplicate-entries
            – mac-table
                – all
                – mac <address>
                – summary
            – proxy-arp
                – all
                – ip-duplication duplicate-entries
                – neighbor <address>
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
            – summary
        –proxy-nd
            – all
            – ip-duplication duplicate-entries
            – neighbor <address>
            – summary
        –stp
            – all
            – interface
            – summary
    – interfaces <interface name>
    – ipv4
        – route
            – all
            – internal-tag
            – leakable
            – leaked
            – next-hop
            – or-longer
            – type type
            – detail
            – summary
    – ipv6
        – route
            – all
            – internal-tag
            – leakable
            – leaked
            – next-hop
            – or-longer
            – type type
            – detail
            – summary
    – multicast-forwarding-information-base
        –ipv4-multicast
        –ipv6-multicast
    – protocols
        – bgp
            – bmp
                – station
                        – statistics
            – malformed-updates
                – summary
            – neighbor [<IP-address>]
                – advertised-routes
                    – evpn
                    – ipv4
                    – ipv4-flowspec-v1
                    – ipv4-labeled-unicast
                    – ipv4-mvpn
                    – ipv6
                    – ipv6-flowspec-v1
                    – ipv6-labeled-unicast
                    – ipv6-mvpn
                    – link-state
                    – l3vpn-ipv4-unicast
                    – l3vpn-ipv6-unicast
                    – route-target
                    – srte-policy-ipv4
                    – srte-policy-ipv6
                – afi-safi name
                    – bmp
                – detail
                – maintenance
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
                    - malformed-updates
                        - summary [detail]
                    - received-routes
                        - evpn
                        - ipv4
                            - origin-validation-state
                        - ipv4-flowspec-v1
                        - ipv4-labeled-unicast
                            - origin-validation-state
                        - ipv4-mvpn
                        - ipv6
                            - origin-validation-state
                        - ipv6-flowspec-v1
                        - ipv6-labeled-unicast
                            - origin-validation-state
                        - ipv6-mvpn
                        - link-state
                        - l3vpn-ipv4-unicast
                        - l3vpn-ipv6-unicast
                        - route-target
                        - srte-policy-ipv4
                        - srte-policy-ipv6
                - routes [<IP family>]
                    - evpn
                        - route-type <route type>
                            - detail
                            - summary
                    - ipv4
                        - origin-validation-state
                        - prefix <IP prefix>
                        - summary
                    - ipv4-flowspec-v1
                        - detail
                        - nlri-filter [detail]
                        - summary
                    - ipv4-labeled-unicast
                        - origin-validation-state
                        - prefix <IP prefix>
                        - summary
                    - ipv4-mvpn
                        - route-type
                            - detail
                            - summary
                    - ipv6
                        - origin-validation-state
                        - prefix <IP prefix>
                        - summary
                    - ipv6-flowspec-v1
                        - detail
                        - nlri-filter [detail]
                        - summary
                    - ipv6-labeled-unicast
                        - origin-validation-state
                        - prefix <IP prefix>
                        - summary
                    - ipv6-mvpn
                        - route-type
                            - detail
                            - summary
                    - link-state
                        - nlri-type
                            - detail
                            - summary
                    - l3vpn-ipv4-unicast
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
                    - summary
                - l3vpn-ipv6-unicast
                    - summary
            - rpki
                - roa-database
                    - ipv4
                        - entry-type
                        - session
                    - ipv6
                        - entry-type
                        - session
                    - origin-as
                    - prefix
                        - longer
                    - session
                        - detail
                        - roa-database
                            - ipv4
                            - ipv6
                        - statistics
            - rpki-ov-statistics
            - summary
        - bgp-evpn
            - bgp-instance
        - bgp-vpn
            - bgp-instance
        - igmp
            - group
            - interface
            - statistics
            - status
        - igmp-snooping
            - evpn-proxy-membership
            - group
            - interface
            - multicast-router
            - proxy-membership
            - statistics
            - status
            - vxlan-destination
        - isis
            - adjacency <interface_name>
                - neighbor-system-id <ID>
                    - adjacency-level <value>
                        - detail
            - database
                - lsp-id <ID>
                - detail
            - interface <name>
                - detail
            - hostnames
                - detail
            - summary
        - ldp
            - interface
            - ipv4
                - address
                    - advertised
                    - all
                    - received
                - fec
                    - advertised
                    - all
                    - received
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
                    - fec128
                        - advertised
                        - all
                        - received
                    - nexthop
                - ipv6
                    - address
                        - advertised
                        - all
                        - received
                    - fec
                        - advertised
                        - all
                        - received
                    - nexthop
                - neighbor
                - session
                    - detail
                    - statistics
                - statistics
                - summary
            - mld
                - group
                - interface
                - statistics
                - status
            - mld-snooping
                - evpn-proxy-membership
                - group
                - interface
                - multicast-router
                - proxy-membership
                - statistics
                - status
                - vxlan-destination
            - msdp
                - group
                - peer
                - source
                - source-active
                - source-active-rejected
                - statistics [<peer>]
                - status
            - ospf
                - area
                    - detail
                - database
                    - type
                - instance <name>
                - interface
                    - detail
                - neighbor
                    - detail
                - statistics
                - status
            - pim
                - database
                - interface
                - neighbor
                - statistics
                - status
            - route-table
                - mpls
        - static-mpls
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
        – summary
        – tunnel-table
            – all
            – ipv4 ip address type
            – ipv6 ip address type
        – vxlan-interface <name>
```

16.8.1 Network-instance descriptions

Context	Description
show network-instance *name*	Show network-instance for specified name or additional command (bridge-table, interfaces, protocols, route-table, or summary).
show network-instance *name* **bridge-table**	Show MAC bridge tables. Specify mac-duplication duplication-entries or mac-table.
show network-instance *name* **bridge-table mac-duplication duplicate-entries**	Show all MAC learned entries.
show network-instance *name* **bridge-table mac-table**	Show a MAC table report. Specify option of all, mac, or summary.
show network-instance *name* **bridge-table mac-table all**	Show all MAC table entries.
show network-instance *name* **bridge-table mac-table mac** *address*	Show MAC table entry for specific address.
show network-instance *name* **bridge-table mac-table summary**	Show a MAC table summary report.
show network-instance *name* **bridge-table proxy-arp**	Show proxy ARP entries report.
show network-instance *name* **bridge-table proxy-arp all**	Show all proxy ARP table entries.
show network-instance *name* **bridge-table proxy-arp ip-duplication duplicate-entries**	Show proxy ARP IP duplication report.
show network-instance *name* **bridge-table proxy-arp neighbor** <*address*>	Show proxy ARP IP table entry.
show network-instance *name* **bridge-table proxy-arp summary**	Show proxy ARP IP table summary.
show network-instance *name* **bridge-table proxy-nd**	Show proxy ND entries report.
show network-instance *name* **bridge-table proxy-nd all**	Show all proxy ND table entries.
show network-instance *name* **bridge-table proxy-nd ip-duplication duplicate-entries**	Show proxy ND IP duplication report.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **bridge-table proxy-nd neighbor** <*address*>	Show proxy ND table entry.
show network-instance *name* **bridge-table proxy-nd summary**	Show proxy ND table summary.
show network-instance *name* **bridge-table stp**	Container for STP reports.
show network-instance *name* **bridge-table stp all**	Show all STP entries.
show network-instance *name* **bridge-table stp interface** <*name*>	Show STP table entry.
show network-instance *name* **bridge-table stp summary**	Show STP summary.
show network-instance *name* **interfaces** [*interface name*]	Show all network-instance interfaces.
show network-instance *name* **ipv4**	Container for IPv4 reports.
show network-instance *name* **ipv4 route [***ipv4-address/ipv4-prefix***]**	Show full IPv4 route table report.
show network-instance *name* **ipv4 route all**	Show all routes for each prefix.
show network-instance *name* **ipv4 route gbp-tag** *tag*	Show routes that contain the specified GBP tag.
show network-instance *name* **ipv4 route internal-tag** *tag*	Show routes that contain the specified internal tag.
show network-instance *name* **ipv4 route leakable**	Show routes that are leakable.
show network-instance *name* **ipv4 route leaked**	Show leaked routes.
show network-instance *name* **ipv4 route next-hop** *next-hop*	Show routes that contain the specified next hop.
show network-instance *name* **ipv4 route or-longer**	Show routes that are a longer match.
show network-instance *name* **ipv4 route type** *type*	Show routes with the specified type.
show network-instance *name* **ipv4 route detail**	Show detailed IPv4 route report.
show network-instance *name* **ipv4 route summary**	Show summary IPv4 route report.
show network-instance *name* **ipv6 route**	Container for IPv6 reports.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **ipv6 route** [*ipv6-address/ipv6-prefix*]	Show full IPv6 route table report.
show network-instance *name* **ipv6 route all**	Show all routes for each prefix.
show network-instance *name* **ipv6 route gbp-tag**	Show routes that contain the specified GBP tag.
show network-instance *name* **ipv6 route internal-tag**	Show routes that contain the specified internal tag.
show network-instance *name* **ipv6 route leakable**	Show routes that are leakable.
show network-instance *name* **ipv6 route leaked**	Show leaked routes.
show network-instance *name* **ipv6 route next-hop**	Show routes that contain the specified next hop.
show network-instance *name* **ipv6 route or-longer**	Show routes that are a longer match.
show network-instance *name* **ipv6 route type** *type*	Show routes with the specified type.
show network-instance *name* **ipv6 route detail**	Show detailed IPv6 route report.
show network-instance *name* **ipv6 route summary**	Show summary IPv6 route report.
show network-instance *name* **multicast-forwarding-information-base**	Show multicast FIB report.
show network-instance *name* **multicast-forwarding-information-base ipv4-multicast**	Show IPv4 multicast FIB report.
show network-instance *name* **multicast-forwarding-information-base ipv6-multicast**	Show IPv6 multicast FIB report.
show network-instance *name* **protocols**	Show network-instance protocol details. Specify BGP, IS-IS, or OSPF.
show network-instance *name* **protocols bgp**	Show BGP status report. Specify neighbor, routes, or summary.
show network-instance *name* **protocols bgp bmp**	Show a summary of the BMP configuration and operational state for a network instance.
show network-instance *name* **protocols bgp bmp station**	Show detailed information about a specific BMP station.
show network-instance *name* **protocols bgp bmp station statistics**	Show detailed statistics for a specific BMP station.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **protocols bgp malformed-updates [afi-safi** *value***] [prefix** *value***] [detail]**	Show retained malformed UPDATE entries across all neighbors
show network-instance *name* **protocols bgp malformed-updates summary**	Show count of retained malformed UPDATE entries per neighbor and AFI-SAFI
show network-instance *name* **protocols bgp neighbor** *IP address*	Show a BGP neighbor report. Specify a peer address to see a report for the specified address.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes**	Show a BGP neighbor advertised routes report. The network-instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes evpn**	Show a BGP neighbor advertised routes report for EVPN. The network-instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes ipv4**	Show a BGP neighbor advertised routes report for IPv4. The network-instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes ipv4-flowspec-v1**	Show a BGP neighbor advertised routes report for IPv4 Flowspec v1.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes ipv4-labeled-unicast**	Show a BGP neighbor advertised routes report for IPv4 labeled unicast. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes ipv4-mvpn**	Show a BGP neighbor advertised routes report for IPv4 multicast VPN. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes ipv6**	Show a BGP neighbor advertised routes report for IPv6. The network-instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes ipv6-flowspec-v1**	Show a BGP neighbor advertised routes report for IPv6 Flowspec v1.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes ipv6-labeled-unicast**	Show a BGP neighbor advertised routes report for IPv6 labeled unicast. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes ipv6-mvpn**	Show a BGP neighbor advertised routes report for IPv6 multicast VPN. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes link-state**	Show a BGP neighbor report for advertised link-state routes. The network instance name and neighbor peer IP address must be specified.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes l3vpn-ipv4-unicast**	Show a BGP neighbor advertised routes report for IPv4 Layer 3 VPN. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes l3vpn-ipv6-unicast**	Show a BGP neighbor advertised routes report for IPv6 Layer 3 VPN. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes route-target**	Show a BGP neighbor advertised routes report for route targets. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes srte-policy-ipv4**	Show a BGP neighbor advertised routes report for IPv4 SR-MPLS TE policy. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **afi-safi** *name* **bmp**	Show the BMP monitoring status for a specific BGP neighbor AFI-SAFI.
show network-instance *name* **protocols bgp neighbor** *IP address* **advertised-routes srte-policy-ipv6**	Show a BGP neighbor advertised routes report for IPv6 SR-MPLS TE policy. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **detail**	Show a BGP neighbor detailed report. network-instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **maintenance**	Show a BGP neighbor report that shows the maintenance mode and group.
show network-instance *name* **protocols bgp neighbor** *IP address* **malformed-updates**	Show malformed UPDATE entries for the specified neighbor.
show network-instance *name* **protocols bgp neighbor** *IP address* **malformed-updates summary [detail]**	Show a summary of retained entries for the specified neighbor. Include the **detail** parameter to list all AFI-SAFIs, including those with no retained entries.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes**	Show a BGP neighbor received routes report. network-instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes evpn**	Show a BGP neighbor received routes report for EVPN. The network-instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv4**	Show a BGP neighbor received routes report for IPv4. The network-instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv4 origin-validation-state** *state*	Restrict output to routes with the specified RPKI origin-validation state.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv4-flowspec-v1**	Show a BGP neighbor received routes report for IPv4 Flowspec v1.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv4-labeled-unicast**	Show a BGP neighbor received routes report for IPv4 labeled unicast. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv4-labeled-unicast origin-validation-state** *state*	Restrict output to routes with the specified RPKI origin-validation state.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv4-mvpn**	Show a BGP neighbor received routes report for IPv4 multicast VPN. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv6**	Show a BGP neighbor received routes report for IPv6. The network-instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv6 origin-validation-state** *state*	Restrict output to routes with the specified RPKI origin-validation state.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv6-flowspec-v1**	Show a BGP neighbor received routes report for IPv6 Flowspec v1.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv6-labeled-unicast**	Show a BGP neighbor received routes report for IPv6 labeled unicast. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv6-labeled-unicast origin-validation-state** *state*	Restrict output to routes with the specified RPKI origin-validation state.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes ipv6-mvpn**	Show a BGP neighbor received routes report for IPv6 multicast VPN. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes link-state**	Show a BGP neighbor report for received link-state routes. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes l3vpn-ipv4-unicast**	Show a BGP neighbor received routes report for IPv4 Layer 3 VPN. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes l3vpn-ipv6-unicast**	Show a BGP neighbor received routes report for IPv6 Layer 3 VPN. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes route-target**	Show a BGP neighbor received routes report for route targets. The network instance name and neighbor peer IP address must be specified.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes srte-policy-ipv4**	Show a BGP neighbor received routes report for IPv4 SR-MPLS TE policy. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp neighbor** *IP address* **received-routes srte-policy-ipv6**	Show a BGP neighbor received routes report for IPv6 SR-MPLS TE policy. The network instance name and neighbor peer IP address must be specified.
show network-instance *name* **protocols bgp routes** *IP family*	Show BGP route report for an IP family (EVPN, IPv4, or IPv6).
show network-instance *name* **protocols bgp routes evpn**	Show a BGP EVPN route report. Additional parameters can define a report for a specific route type, and a detailed or summary report.
show network-instance *name* **protocols bgp routes evpn route-type** *route type*	Show a BGP EVPN route report for a specific route type. Route type can be numeric or of type esi, mac-address, and originating-router.
show network-instance *name* **protocols bgp routes evpn route-type** *route type* **detail**	Show a detailed BGP EVPN route report for a specific route type.
show network-instance *name* **protocols bgp routes evpn route-type** *route type* **summary**	Show a summarized BGP EVPN route report for a specific route type.
show network-instance *name* **protocols bgp routes ipv4**	Show a BGP IPv4 route report. Specify an IP prefix or summary.
show network-instance *name* **protocols bgp routes ipv4-flowspec-v1**	Show an IPv4 BGP FlowSpec-v1 route report. Specify detail, an NLRI filter, or summary.
show network-instance *name* **protocols bgp routes ipv4 prefix** *IP prefix*	Show a BGP IPv4 route report for a specified IP prefix.
show network-instance *name* **protocols bgp routes ipv4 summary**	Show a BGP IPv4 route summary report.
show network-instance *name* **protocols bgp routes ipv6**	Show a BGP IPv6 route report. Specify an IP prefix or summary.
show network-instance *name* **protocols bgp routes ipv6 origin-validation-state** *state*	Restrict output to routes with the specified RPKI origin-validation state.
show network-instance *name* **protocols bgp routes ipv6 prefix** *IP prefix*	Show a BGP IPv6 route report for a specified IP prefix.
show network-instance *name* **protocols bgp routes ipv6 summary**	Show a BGP IPv6 route summary report.
show network-instance *name* **protocols bgp routes ipv4-labeled-unicast**	Show a BGP IPv4 labeled unicast route report. Specify an IP prefix or summary.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **protocols bgp routes ipv4-labeled-unicast origin-validation-state** *state*	Restrict output to routes with the specified RPKI origin-validation state.
show network-instance *name* **protocols bgp routes ipv4-labeled-unicast prefix** *IP prefix*	Show a BGP IPv4 labeled unicast route report for a specified IP prefix.
show network-instance *name* **protocols bgp routes ipv4-labled-unicast summary**	Show a BGP IPv4 labeled unicast route summary report.
show network-instance *name* **protocols bgp routes ipv4-mvpn route-type** *route-type*	Show a BGP IPv4 multicast VPN route report for a specific route type. Route type can be I-PMSI AD, leaf AD, shared-tree-join, source-active AD, source-tree-join, or S-PMSI AD.
show network-instance *name* **protocols bgp routes ipv4-mvpn route-type** *route-type* **detail**	Show a detailed BGP IPv4 multicast VPN route report for a specific route type.
show network-instance *name* **protocols bgp routes ipv4-mvpn route-type** *route-type* **summary**	Show a summarized BGP IPv4 multicast VPN route report for a specific route type.
show network-instance *name* **protocols bgp routes ipv6-flowspec-v1**	Show a BGP IPv6 FlowSpec-v1 route report. Specify detail, an NLRI filter, or summary.
show network-instance *name* **protocols bgp routes ipv6-labeled-unicast**	Show a BGP IPv6 labeled unicast route report. Specify an IP prefix or summary.
show network-instance *name* **protocols bgp routes ipv6-labeled-unicast origin-validation-state** *state*	Restrict output to routes with the specified RPKI origin-validation state.
show network-instance *name* **protocols bgp routes ipv6-labeled-unicast prefix** *IP prefix*	Show a BGP IPv6 labeled unicast route report for a specified IP prefix.
show network-instance *name* **protocols bgp routes ipv6-labeled-unicast summary**	Show a BGP IPv6 labeled unicast route summary report.
show network-instance *name* **protocols bgp routes ipv6-mvpn route-type** *route-type*	Show a BGP IPv6 multicast VPN route report for a specific route type. Route type can be I-PMSI AD, leaf AD, shared-tree-join, source-active AD, source-tree-join, or S-PMSI AD.
show network-instance *name* **protocols bgp routes ipv6-mvpn route-type** *route-type* **detail**	Show a detailed BGP IPv6 multicast VPN route report for a specific route type.
show network-instance *name* **protocols bgp routes ipv6-mvpn route-type** *route-type* **summary**	Show a summarized BGP IPv6 multicast VPN route report for a specific route type.
show network-instance *name* **protocols bgp routes link-state nlri-type** *nlri-type*	Show a BGP link-state report for a specific NLRI type. NLRI type can be IPv4 topology prefix, IPv6 topology prefix, link, node, or SRv6 SID.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **protocols bgp routes link-state nlri-type** *nlri-type* **detail**	Show a detailed BGP link-state report for a specific NLRI type.
show network-instance *name* **protocols bgp routes link-state nlri-type** *nlri-type* **summary**	Show a summarized BGP IPv6 multicast VPN route report for a specific route type.
show network-instance *name* **protocols bgp routes l3vpn-ipv4-unicast**	Show a BGP IPv4 unicast routes report for Layer 3 VPN.
show network-instance *name* **protocols bgp routes l3vpn-ipv4-unicast summary**	Show a summary report of BGP IPv4 unicast routes for Layer 3 VPN.
show network-instance *name* **protocols bgp routes l3vpn-ipv6-unicast**	Show a BGP IPv6 unicast routes report for Layer 3 VPN.
show network-instance *name* **protocols bgp routes l3vpn-ipv4-unicast summary**	Show a summary report of BGP IPv6 unicast routes for Layer 3 VPN.
show network-instance *name* **protocols bgp rpki**	Show RPKI origin-validation state
show network-instance *name* **protocols bgp rpki roa-database**	Show the RPKI ROA database
show network-instance *name* **protocols bgp rpki roa-database ipv4**	Show IPv4 ROA entries across all cache-server sessions and static ROAs
show network-instance *name* **protocols bgp rpki roa-database ipv4 entry-type** *type*	Restrict output by ROA entry source classification
show network-instance *name* **protocols bgp rpki roa-database ipv4 session** *name*	Restrict the output to a single RPKI-RTR session
show network-instance *name* **protocols bgp rpki roa-database ipv6**	Show IPv6 ROA entries across all cache-server sessions and static ROAs
show network-instance *name* **protocols bgp rpki roa-database ipv6 entry-type** *name*	Restrict output by ROA entry source classification
show network-instance *name* **protocols bgp rpki roa-database ipv6 session** *name*	Restrict the output to a single RPKI-RTR session
show network-instance *name* **protocols bgp rpki roa-database origin-as** *asn*	Filter ROA entries by origin AS number
show network-instance *name* **protocols bgp rpki roa-database prefix**	Filter ROA entries by prefix
show network-instance *name* **protocols bgp rpki roa-database prefix longer**	Include equal and more-specific prefixes within the given prefix

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **protocols bgp rpki session** *name*	Show RPKI-RTR cache-server sessions
show network-instance *name* **protocols bgp rpki session** *name* **detail**	Show detailed RPKI-RTR session state and statistics
show network-instance *name* **protocols bgp rpki session** *name* **roa-database**	Show the ROA entries received from this session
show network-instance *name* **protocols bgp rpki session** *name* **roa-database ipv4**	Show IPv4 ROA entries received from this session
show network-instance *name* **protocols bgp rpki session** *name* **roa-database ipv6**	Show IPv6 ROA entries received from this session
show network-instance *name* **protocols bgp rpki session** *name* **statistics**	Show RPKI-RTR session PDU and error counters
show network-instance *name* **protocols bgp rpki-ov-statistics [detail]**	Show RPKI origin-validation statistics
show network-instance *name* **protocols bgp summary**	Show a BGP summary report.
show network-instance *name* **protocols bgp-evpn**	Show a BGP-EVPN status report.
show network-instance *name* **protocols bgp-evpn bgp-instance** *instance*	Show a BGP-EVPN status report for a specific bgp instance.
show network-instance *name* **protocols bgp-vpn**	Show a BGP-VPN status report.
show network-instance *name* **protocols bgp-vpn bgp-instance** *instance*	Show a BGP-VPN status report for a specific bgp instance.
show network-instance *name* **protocols igmp**	Container for IGMP show reports.
show network-instance *name* **protocols igmp group [** *interface-name***] [group** *value***] [detail]**	Show the IGMP report for an IGMP group.
show network-instance *name* **protocols igmp interface [***interface-name***] [detail]**	Show the IGMP report for an interface.
show network-instance *name* **protocols igmp statistics**	Show the IGMP statistics report.
show network-instance *name* **protocols igmp status**	Show the IGMP status report.
show network-instance *name* **protocols igmp-snooping**	Container for IGMP snooping show reports.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **protocols igmp-snooping evpn-proxy-membership [group** *value***] [detail]**	Show the IGMP snooping report for EVPN proxy membership.
show network-instance *name* **protocols igmp-snooping group** *[interface-name]* **[group** *value***] [detail]**	Show the IGMP snooping group report.
show network-instance *name* **protocols igmp-snooping interface** *[interface-name]* **[detail]**	Show the IGMP snooping interface report.
show network-instance *name* **protocols igmp-snooping multicast-router [address** *value***] [detail]**	Show the IGMP snooping multicast router report.
show network-instance *name* **protocols igmp-snooping proxy-membership [group** *value***] [detail]**	Show the IGMP snooping proxy membership report.
show network-instance *name* **protocols igmp-snooping statistics** *[interface-name]*	Show the IGMP snooping statistics report.
show network-instance *name* **protocols igmp-snooping status**	Show the IGMP snooping status report.
show network-instance *name* **protocols igmp-snooping vxlan-destination [***vtep***] [***vni***] [detail]**	Show the IGMP snooping VXLAN destination report.
show network-instance *name* **protocols isis**	Show IS-IS status report. Specify adjacency, database, hostnames, interface, or summary.
show network-instance *name* **protocols isis adjacency** *interface_name*	Show a list of IS-IS adjacencies formed through this interface.
show network-instance *name* **protocols isis adjacency** *interface_name* **neighbor-system-id** *ID*	Show a list of IS-IS adjacencies for a specified system ID of a neighbor router.
show network-instance *name* **protocols isis adjacency** *interface_name* **neighbor-system-id** *ID* **adjacency-level** *value*	Show a list of IS-IS adjacencies for a specified adjacency level.
show network-instance *name* **protocols isis adjacency** *interface_name* **neighbor-system-id** *ID* **adjacency-level** *value* **detail**	Show a detailed IS-IS adjacency report.
show network-instance *name* **protocols isis database**	Show an IS-IS database report.
show network-instance *name* **protocols isis database lsp-id** *ID*	Show an IS-IS database report for a specified lsp-id (ID format: 6 octets of adjacency system-id followed by 1 octet Lan-ID and 1 octet LSP Number).
show network-instance *name* **protocols isis database detail**	Show a detailed IS-IS database report.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **protocols isis interface** *name*	Show an IS-IS interface report for all interfaces or optionally specify an interface name.
show network-instance *name* **protocols isis interface** *name* **detail**	Show a detailed IS-IS interface report for all interfaces or optionally specify an interface name.
show network-instance *name* **protocols isis hostnames**	Show an IS-IS hostname report.
show network-instance *name* **protocols isis hostnames detail**	Show a detailed IS-IS hostname report.
show network-instance *name* **protocols isis summary**	Show a summarized IS-IS report.
show network-instance *name* **protocols ldp**	Container for LDP show reports.
show network-instance *name* **protocols ldp interface** *name* **[detail]**	Show the LDP report for an interface.
show network-instance *name* **protocols ldp ipv4**	Container for IPv4 LDP reports.
show network-instance *name* **protocols ldp ipv4 address**	Show the LDP report for IPv4 addresses.
show network-instance *name* **protocols ldp ipv4 address advertised [table| summary] [lsr-id** *value***] [label-space-id***value***]**	Show the LDP report for advertised IPv4 addresses.
show network-instance *name* **protocols ldp ipv4 address all [table|summary] [lsr-id** *value***] [label-space-id***value***]**	Show the LDP report for all IPv4 addresses.
show network-instance *name* **protocols ldp ipv4 address received [table| summary] [lsr-id** *value***] [label-space-id***value***]**	Show the LDP report for received IPv4 addresses.
show network-instance *name* **protocols ldp ipv4 fec**	Show the LDP report for IPv4 FECs.
show network-instance *name* **protocols ldp ipv4 fec advertised [table|summary| detail] [prefix** *value* **[lsr-id** *value***] [label-space-id** *value***]**	Show the LDP report for advertised IPv4 FECs.
show network-instance *name* **protocols ldp ipv4 fec all [table|summary|detail] [prefix** *value* **[lsr-id** *value***] [label-space-id** *value***]**	Show the LDP report for all IPv4 FECs.
show network-instance *name* **protocols ldp ipv4 fec received [table|summary|**	Show the LDP report for received IPv4 FECs.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
detail] [prefix *value* **[lsr-id** *value*] **[label-space-id** *value*]	
show network-instance *name* **protocols ldp ipv4 fec128**	Show the LDP report for FEC 128 IPv4 FECs.
show network-instance *name* **protocols ldp ipv4 fec128 advertised [table| summary|detail] [prefix** *value* **[lsr-id** *value*] **[label-space-id** *value*]	Show the LDP report for advertised FEC 128 IPv4 FECs.
show network-instance *name* **protocols ldp ipv4 fec128 all [table|summary| detail] [prefix** *value* **[lsr-id** *value*] **[label-space-id** *value*]	Show the LDP report for all FEC 128 IPv4 FECs.
show network-instance *name* **protocols ldp ipv4 fec128 received [table| summary|detail] [prefix** *value* **[lsr-id** *value*] **[label-space-id** *value*]	Show the LDP report for received FEC 128 IPv4 FECs.
show network-instance *name* **protocols ldp ipv4 nexthop [fec-prefix** *value*] **[lsr-id** *value*] **[label-space-id** *value*]	Show the LDP report for IPv4 next-hops.
show network-instance *name* **protocols ldp ipv6**	Container for IPv6 LDP reports.
show network-instance *name* **protocols ldp ipv6 address**	Show the LDP report for IPv6 addresses.
show network-instance *name* **protocols ldp ipv6 address advertised [table| summary] [lsr-id** *value*] **[label-space-id** *value*]	Show the LDP report for advertised IPv6 addresses.
show network-instance *name* **protocols ldp ipv6 address all [table|summary] [lsr-id** *value*] **[label-space-id** *value*]	Show the LDP report for all IPv6 addresses.
show network-instance *name* **protocols ldp ipv6 address received [table| summary] [lsr-id** *value*] **[label-space-id** *value*]	Show the LDP report for received IPv6 addresses.
show network-instance *name* **protocols ldp ipv6 fec**	Show the LDP report for IPv6 FECs.
show network-instance *name* **protocols ldp ipv6 fec advertised [table|summary| detail] [prefix** *value* **[lsr-id** *value*] **[label-space-id** *value*]	Show the LDP report for advertised IPv6 FECs.
show network-instance *name* **protocols ldp ipv6 fec all [table|summary|detail]**	Show the LDP report for all IPv6 FECs.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
[prefix *value* **[lsr-id** *value***] [label-space-id** *value***]**	
show network-instance *name* **protocols ldp ipv6 fec received [table|summary| detail] [prefix** *value* **[lsr-id** *value***] [label-space-id** *value***]**	Show the LDP report for received IPv6 FECs.
show network-instance *name* **protocols ldp ipv6 nexthop [fec-prefix** *value* **[lsr-id** *value***] [label-space-id** *value***]**	Show the LDP report for IPv6 next-hops.
show network-instance *name* **protocols ldp neighbor**	Show the LDP neighbor report.
show network-instance *name* **protocols ldp session [lsr-id** *value***] [label-space-id** *value***]**	Show the LDP session report.
show network-instance *name* **protocols ldp session [lsr-id** *value***] [label-space-id** *value***] detail**	Show the detailed LDP session report.
show network-instance *name* **protocols ldp session [lsr-id** *value***] [label-space-id** *value***] statistics**	Show the LDP session statistics report.
show network-instance *name* **protocols ldp statistics**	Show the LDP statistics report.
show network-instance *name* **protocols ldp summary**	Show the LDP summary report.
show network-instance *name* **protocols mld**	Container for MLD reports.
show network-instance *name* **protocols mld group [** *interface-name***] [group** *value***] [detail]**	Show the MLD group report.
show network-instance *name* **protocols mld interface [***interface-name***] [detail]**	Show the MLD interface report.
show network-instance *name* **protocols mld statistics**	Show the MLD statistics report.
show network-instance *name* **protocols mld status**	Show the MLD status report.
show network-instance *name* **protocols mld-snooping**	Container for MLD snooping show reports.
show network-instance *name* **protocols mld-snooping evpn-proxy-membership [group** *value***] [detail]**	Show the MLD snooping report for EVPN proxy membership.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **protocols mld-snooping group** *[interface-name]* **[group** *value***] [detail]**	Show the MLD snooping group report.
show network-instance *name* **protocols mld-snooping interface** *[interface-name]* **[detail]**	Show the MLD snooping interface report.
show network-instance *name* **protocols mld-snooping multicast-router [address** *value***] [detail]**	Show the MLD snooping multicast router report.
show network-instance *name* **protocols mld-snooping proxy-membership [group** *value***] [detail]**	Show the MLD snooping proxy membership report.
show network-instance *name* **protocols mld-snooping statistics** *[interface-name]*	Show the MLD snooping statistics report.
show network-instance *name* **protocols mld-snooping status**	Show the MLD snooping status report.
show network-instance *name* **protocols mld-snooping vxlan-destination [**_vtep_**] [**_vni_**] [detail]**	Show the MLD snooping VXLAN destination report.
show network-instance *name* **protocols msdp**	Container for MSDP reports.
show network-instance *name* **protocols msdp group [group-name** *value***] [detail]**	Show the MSDP group report.
show network-instance *name* **protocols msdp peer [peer-address** *value* **] [group** *value***] [detail]**	Show the MSDP peer report.
show network-instance *name* **protocols msdp source [ip-prefix** *value* **] [detail]**	Show the MSDP source report.
show network-instance *name* **protocols msdp source-active [group** *value* **] [source** *value***] [origin-rp** *value***] [detail]**	Show the MSDP source-active report.
show network-instance *name* **protocols msdp source-active-rejected [group-name** *value***] [group** *value***] [source** *value* **] [origin-rp** *value* **] [peer** *value***] [detail]**	Show the MSDP source-active-rejected report.
show network-instance *name* **protocols msdp statistics [**_peer_**]**	Show the MSDP statistics report.
show network-instance *name* **protocols msdp status**	Show the MSDP status report.
show network-instance *name* **protocols ospf**	Show OSPF status report. Specify area, interface, neighbor, or status.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **protocols ospf area**	Show OSPF report for areas where the local system exists.
show network-instance *name* **protocols ospf area detail**	Show detailed OSPF report for areas where the local system exists.
show network-instance *name* **protocols ospf database**	Show OSPF database report.
show network-instance *name* **protocols ospf database type**	Show OSPF database report for a specific database type. For example: router-lsa
show network-instance *name* **protocols ospf instance** *instance name*	Show OSPF report for a specific instance. After the instance name, a more detailed report can be generated by specifying area, database, interface, neighbor, statistics, or status.
show network-instance *name* **protocols ospf interface**	Show OSPF report for OSPF interfaces that act as a connection between a router and one of its attached networks.
show network-instance *name* **protocols ospf interface detail**	Show detailed OSPF report for OSPF interfaces that act as a connection between a router and one of its attached networks.
show network-instance *name* **protocols ospf neighbor**	Show OSPF report for OSPF neighbors.
show network-instance *name* **protocols ospf neighbor detail**	Show detailed OSPF report for OSPF neighbors.
show network-instance *name* **protocols ospf statistics**	Show OSPF Rx, Tx, and packet statistics report.
show network-instance *name* **protocols ospf status**	Show an overall status report for OSPF (router IDs, version, admin state, backbone and area border router settings, and so on).
show network-instance *name* **protocols pim**	Container for PIM show reports.
show network-instance *name* **protocols pim database [ipv4] [ipv6] [group** *value***] [source** *value***] [detail]**	Show the PIM database report.
show network-instance *name* **protocols pim interface** *[interface-name]* **[ipv4] [ipv6] [detail]**	Show the PIM interface report.
show network-instance *name* **protocols pim neighbor** *[interface-name]* **[ipv4] [ipv6] [detail]**	Show the PIM neighbor report.
show network-instance *name* **protocols pim statistics** *[interface-name]* **[ipv4] [ipv6]**	Show the PIM statistics report.
show network-instance *name* **protocols pim status**	Show the PIM status report.
show network-instance *name* **route-table**	Show a network-instance route table report.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show network-instance *name* **route-table mpls**	Show MPLS table report that includes labels, next-hop IP, next-hop subinterface, and next-hop labels.
show network-instance *name* **summary**	Show a summary report for all network-instances, or specify a network-name to refine the report.
show network-instance *name* **static-mpls**	Show static MPLS entries.
show network-instance *name* **tunnel-table**	Context for all tunnels in a tunnel table. Use with options all, IPv4, or IPv6.
show network-instance *name* **tunnel-table all**	Shows a network instance tunnel report that includes the prefix, type, ID, metric, preference, and last updated fields.
show network-instance *name* **tunnel-table ipv4 ip address type**	Show a network instance IPv4 tunnel report that includes the prefix, type, ID, metric, preference, and last updated fields. Report can be defined further by specifying the type (such as vxlan, ldp, * [all types]).
show network-instance *name* **tunnel-table ipv6 ip address type**	Show a network instance IPv6 tunnel report that includes the prefix, type, ID, metric, preference, and last updated fields. Report can be defined further by specifying the type (such as vxlan, ldp, and * [all types]).
show network-instance *name* **vxlan-inteface** *name*	Show a network instance vxlan-interface report that defines the type, operation state and operation down reason for a specified interface or all interfaces (when the vxlan-interface name is "*").

16.9 OAM show reports

```
show
    – oam
        – link-measurement
            – interface
                – aggregate
                – debug-counters
                – detail
                – lag-micro-sessions
                    – port
                        – aggregate
                        – debug-counters
                        – detail
                        – sample
                – sample
        – measurement-template
    – service-activation-testhead
        – service-activation-tests
        – service-test
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

16.9.1 OAM descriptions

Context	Description
show oam	Container for OAM reports.
show oam link-measurement	Show link measurement templates information.
show oam link-measurement interface	Show summary when name is omitted or *; specific interface name shows measurement fields; **detail** adds sample/aggregate windows; **sample**, **aggregate**, or **debug-counters** for window tables.
show oam link-measurement interface aggregate	Show aggregate sample window statistics.
show oam link-measurement interface debug-counters	Show sample window debug counters.
show oam link-measurement interface detail	Show interface measurements with all sample and aggregate windows.
show oam link-measurement interface lag-micro-sessions	Show LAG micro-session information for all member ports (no window statistics).
show oam link-measurement interface lag-micro-sessions port	Show reports for LAG member port.
show oam link-measurement interface lag-micro-sessions port aggregate	Show LAG member aggregate sample window statistics.
show oam link-measurement interface lag-micro-sessions port debug-counters	Show LAG member sample window debug counters.
show oam link-measurement interface lag-micro-sessions port detail	Show LAG member port summary plus all sample and aggregate windows; with **port *** also shows LAG interface summary for all members.
show oam link-measurement interface lag-micro-sessions port sample	Show LAG member sample window statistics.
show oam link-measurement interface sample	Show sample window statistics.
show oam link-measurement measurement-template	Show link measurement templates information.
show oam service-activation-testhead service-activation-tests	Show a report of service activation tests.
show oam service-activation-testhead service-test *test-name* **run** *value*	Show a report for a specified service test.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

16.10 Platform show reports

```
show
    – platform
        – chassis
        – control <slot ID>
        – environment
        – fabric <slot number>
        – fan-tray <ID number>
        – linecard <slot number>
        – power-supply <ID number>
        – redundancy
        – resource-monitoring
        – trust
            – secure-boot
                – control
                    – detail
            – tpm
                – certificate
                – pcr-bank
```

16.10.1 Platform descriptions

Context	Description
show platform	Show a state report for all components.
show platform chassis	Show a report for the chassis, including type, MAC address, number of slots, and operating status.
show platform control *slot ID*	Show a report for the control module configuration and state. Shows all modules unless a specific slot is provided.
show platform environment	Show a report for the platform environment (state and temperature) for all components.
show platform fabric *slot ID*	Show a report for the fabric module configuration and state. Shows all modules unless a specific slot is provided.
show platform fan-tray *slot ID* **[detail]**	Show a report for the fan tray configuration and state. Shows all modules unless a specific slot is provided.
show platform linecard *slot ID*	Show a report for the linecard configuration and state. Shows all modules unless a specific slot is provided.
show platform power-supply *ID*	Show a report for the power-supply configuration and state. Shows all modules unless a specific ID is provided.
show platform redundancy	Show a platform redundancy report.
show platform resource-monitoring	Show a resource-monitoring report for areas such as ACL, TCAM, IP MPLS forwarding, etc.
show platform trust	Show platform trust information.
show platform trust secure-boot	Show platform secure boot information.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show platform trust secure-boot control *id*	Show secure boot information for control modules.
show platform trust secure-boot control *id* **detail**	Show detailed secure boot information for control modules.
show platform trust tpm *tpm-id*	Show TPM information.
show platform trust tpm *tpm-id* **certificate** *certificate*	Show certificates from TPM NV Memory.
show platform trust tpm *tpm-id* **pcr-bank** *pcr-bank*	Show TPM PCR allocation.

16.11 QoS show reports

```
show
    – qos
        – interfaces
            – mismatches [interface interface-name] [path path]
            – statistics [queue] interface interface-name
```

16.11.1 QoS descriptions

Context	Description
show qos	Container for QoS show reports.
show qos interfaces mismatches [interface *interface-name*] **[path** *path*]	Show QoS mismatches between configuration and state.
show qos statistics [queue] interface *interface-name*	Show QoS interface or subinterface statistics

16.12 System show reports

```
show
    – system
        – aaa authentication session <session ID number>
        – alarms
        – application <application name>
            – lldp neighbor
            – interface <interface name>
        – ipfix
            – exporting-process
                – destination [<name>] [detail]
            – observation-point
                – interface [interface <interface name>]
            – status
        – logging
```


© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

```
            – buffer
                – messages
                – system
            – file
                – messages
        – network-instance <name>
            – ethernet-segments
            – mgmt-stp
                – interface
        – sflow status
        – srv6
            – micro-segment-locator
                – local-sids
```

16.12.1 System descriptions

Context	Description
show system	Container for system management show reports.
show system aaa authentication session *session ID*	Show a list of all active sessions in the system or for a specific session ID user if an ID is specified.
show system alarms	Show alarms information.
show system application *application name*	Show a list of all applications and their status in the system or for a specific application if a name is specified.
show system ipfix	Container for IPFIX show reports.
show system ipfix exporting-processdestination [<*name*>**] [detail]**	Show IPFIX exporting process report.
show system ipfix observation-point interface [<*interface name*>**]**	Show IPFIX observation point report.
show system ipfix status	Show IPFIX status.
show system lldp neighbor	Show a report of all LLDP neighbors that includes system name, chassis ID, first message, last update, port and related BGP details.
show system lldp neighbor interface *interface name*	Show a report of a specific LLDP neighbors by specifying an interface name.
show system logging	Shows a list of system logs, type, and directory path.
show system logging buffer	Shows a report of type buffer log files maintained in memory (non-persistent across system reboots).
show system logging buffer messages	Show a report of the messages associated with the system logging buffer (/var/log/srlinux/buffer/ messages).
show system logging buffer system	Show a report of the system related information in the system logging buffer (/var/log/srlinux/buffer/system).
show system logging file	Shows a report of log files that have been directed to a specific file (/var/log/srlinux/file).

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
show system logging file messages	Show a report of the messages associated with logs in a specific file (/var/log/srlinux/file/messages).
show system network-instance ethernet-segments name	Show a report that defines ethernet-segments details such as ESI, Alg, peers, and associated network instances.
show system srv6	Container for segment routing over IPv6 (SRv6) show reports.
show system srv6 micro-segment-locator	Show an SRv6 report for micro segment locators.
show system srv6 micro-segment-locator local-sids	Show an SRv6 report for local SIDs for micro segment locators.
show system mgmt-stp *<name>*	Show an MSTP report.
show system mgmt-stp *<name>* **interface** *<interface-name>*	Show an MSTP report displaying interface details.
show system sflow status	Show an sflow status report that includes the state, and sample size and rate.

16.13 Tunnel show reports

```
show
    – tunnel
        – vxlan-tunnel
            – all
            – vtep <ip address>
```

16.13.1 Tunnel descriptions

Context	Description
show tunnel	Enter the tunnel context.
show tunnel vxlan-tunnel	Show a tunnel report specific to VXLAN tunnels. Report includes the VXLAN tunnel endpoint address, index and when last updated. Specify a specific VTEP ip address or 'all' to display all vxlan tunnels.
show tunnel vxlan-tunnel all	Show a tunnel report specific to all VXLAN tunnels. Report includes the VXLAN tunnel endpoint address, index, and when last updated.
show tunnel vxlan-tunnel vtep *ip address*	Show a tunnel report for a specific VXLAN tunnel. Report includes the VXLAN tunnel endpoint address, index, and when last updated.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

16.14 Tunnel-interface show reports

```
show
    – tunnel-interface <interface>
        – vxlan-interface <interface>
            – bridge-table
                – mac-table
                – multicast-destinations
                    – destination | VNI
                – unicast-destinations
                    – destination | VNI
            – brief
            – detail
```

16.14.1 Tunnel-interface descriptions

Context	Description
show tunnel-interface *interface*	Enter the context for tunnel interface (reports for EVPN Layer-2). Specify a tunnel-interface name or an asterisk (*) for all interfaces.
show tunnel-interface *interface* **vxlan-interface** *interface*	Show a vxlan interface report for EVPN-VXLAN for layer-2. Specify an vxlan-interface name or an asterisk (*) for all interfaces. Report options include brief, detail, or bridge-table.
show tunnel-interface *interface* **vxlan-interface** *interface* **bridge-table**	Show a bridge-table vxlan interface report for EVPN-VXLAN for layer-2. Options to use with bridge table include mac-table, multicast-destinations, and unicast-destination.
show tunnel-interface *interface* **vxlan-interface** *interface* **bridge-table mac-table**	Show a mac-table vxlan interface report for EVPN-VXLAN for layer-2. Report includes the address, destination, VNI, index, type, and last updated.
show tunnel-interface *interface* **vxlan-interface** *interface* **bridge-table multicast-destinations**	Show a multicast-destinations vxlan interface report for EVPN-VXLAN for layer-2. Use with the destination or VNI options.
show tunnel-interface *interface* **vxlan-interface** *interface* **bridge-table multicast-destinations destination | VNI**	Show a multicast-destinations vxlan interface report for EVPN-VXLAN for layer-2 for a specific destination or VNI or use the asterisk after either option to see all destinations/VNIs.
show tunnel-interface *interface* **vxlan-interface** *interface* **bridge-table unicast-destinations**	Show a unicast-destinations vxlan interface report for EVPN-VXLAN for layer-2. Use with the destination or VNI options.
show tunnel-interface *interface* **vxlan-interface** *interface* **bridge-table unicast-destinations destination | VNI**	Show a unicast-destinations vxlan interface report for EVPN-VXLAN for layer-2 for a specific destination or VNI or use the asterisk after either option to see all destinations/VNIs.
show tunnel-interface *interface* **vxlan-interface** *interface* **brief**	Show a brief vxlan interface report for EVPN-VXLAN for layer-2. The brief report includes the tunnel-interface, VXLAN-

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Context	Description
	interface, type (bridged/routed), ingress VNI, and egress source-ip.
show tunnel-interface *interface* **vxlan-interface** *interface* **detail**	Show a detailed vxlan interface report for EVPN-VXLAN for layer-2. The detailed report includes the same details as the brief report (tunnel-interface, VXLAN-interface, type (bridged/routed), ingress VNI, and egress source-ip), plus bridge table and summary details.

16.15 Version show reports

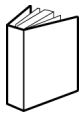
```
show
    – version
```

16.15.1 Version descriptions

Context	Description
show version	Show a version report that contains the currently running software version, last booted, and memory details.

© 2026 Nokia.
Use subject to Terms available at: www.nokia.com/terms.

Customer document and product support



Customer documentation

Customer documentation welcome page



Technical support

Product support portal



Documentation feedback

Customer documentation feedback