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

Release 26.10

gRIBI Guide

3HE 22259 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....4
1.1 Precautionary and information messages....4
1.2 Conventions....4

2 What's new....6

3 About gRIBI....7
3.1 Modify RPC....8
3.1.1 Redundancy and persistence session parameters....8
3.1.2 Election ID....10
3.2 Get RPC....10
3.2.1 Reconciliation....10
3.2.2 Application warm restart....11
3.3 Flush RPC....11
3.4 Authentication....11
3.5 Event Logging....12

4 gRIBI configuration....13
4.1 Enabling gRIBI service support....13
4.2 Configuring a UNIX socket for the gRPC server....14
4.3 Configuring gRIBI protocol parameters for the network instance....15

5 gRIBI display commands....17
5.1 Displaying gRIBI client information....17
5.2 Displaying gRIBI route information....19

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

1 About this guide

This document describes configuration details for the gRPC Routing Information Base Interface (gRIBI) feature set used with the Nokia Service Router Linux (SR Linux).

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

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

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

1.1 Precautionary and information messages

The following are information symbols used in the documentation.

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

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

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

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

1.2 Conventions

The Nokia SR Linux documentation uses the following command conventions:

- **Bold** type indicates a command that the user must enter.
- Input and output examples are displayed in `Courier` text.
- A vertical bar (|) indicates a mutually exclusive argument.
- Square brackets ([ ]) indicate optional elements.

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

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

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.

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

There have been no updates in this document since it was last released.

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

3 About gRIBI

gRIBI (gRPC Routing Information Base Interface) is a gRPC-based protocol that allows external clients to inject routes into the RIB of a network device. The gRIBI clients add and remove RIB entries using a simple API that is defined by the gribi.proto. RIB entries are defined using the OpenConfig Abstract Forwarding Table (AFT) model, translated to protobuf (using ygot). The AFT model allows for a common vendor-independent abstraction of RIB information.

The protobuf definition of gRIBI is maintained in the gRIBI GitHub repository: gribi.proto. The corresponding gribi_aft.proto containing the OpenConfig AFT model is available in the same repository.

The fundamentals of the gRIBI interface are as follows:

- The interface acts as a client of the routing table manager (fib_mgr) on the device.
- Injected gRIBI entries can be interdependent on entries from other protocols, with the device handling resolution. (These entries are published in the RIB view of the system, which allows them to be observed using gNMI.)
- The interface uses transactional semantics with a request/response design, such that the programming entity can determine the success or failure of an operation.
- Rather than inject entries to appear as though they are coming from a particular protocol, gRIBI injects entries separately to live alongside the other existing protocols.
- The gRIBI interface is normalised across vendors. Similar to OpenConfig, gNMI and gNOI, translation to vendor-specific data models is performed on the device, where it can be performed most effectively.
- The interface is fundamentally considered to be part of the control plane of the device, not the management plane. Entries are created as though they are learned through a dynamic routing protocol, not as transient device configurations.

Supported RPCs

SR Linux supports the following gRIBI RPCs:

- Modify RPC (ADD, DELETE, and REPLACE AFTOperations)
- Get RPC
- Flush RPC

Supported AFTs

SR Linux supports the following OpenConfig AFTs:

- IPv4
- Next-Hop Group
- Next-Hop

Supported platforms

gRIBI is supported on the following hardware platforms:

- 7250 IXR-6

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

- 7250 IXR-10
- 7250 IXR-6e
- 7250 IXR-10e
- 7250 IXR-X3b

3.1 Modify RPC

The Modify RPC is the primary gRIBI RPC, allowing gRIBI clients to add, modify, and remove entries in the RIB. The Modify RPC is a bidirectional streaming RPC that modifies the AFT using ModifyRequests. When the gRIBI server completes ModifyRequest operations, it responds (asynchronously) with one or more ModifyResponses indicating what actions were taken.

The structure of the Modify RPC is as follows:

```
rpc Modify(stream ModifyRequest) returns (stream ModifyResponse);
```

Ordering requirements

gRIBI clients must ensure the correct ordering of gRIBI transactions within ModifyRequests. Specifically, the clients must send next-hop and next-hop group entries before any IPv4 entry that depends on them. Clients send next hop groups and next hops as repeated AFTOperation messages in a single ModifyRequest. However, the clients must wait for the next-hop and next-hop group transactions to be acknowledged before proceeding to program the corresponding IPv4 entry.

To enforce the correct transaction ordering, the gRIBI server performs the following error handling:

- If a next-hop group entry references a non-existent next-hop, then the programmed next-hop group is considered invalid and an error is returned to the client.
- During deletion, if an entry to be deleted is still a dependency for another entry (for example, a next-hop that is still referenced from a next-hop group), an error is returned.

3.1.1 Redundancy and persistence session parameters

The gRIBI API allows multiple gRIBI clients to connect to the network device at once. To define how the network device interacts with the client sessions, the Modify RPC provides the following session parameters:

- redundancy: defines whether the network device accepts entries from all clients or from a primary client only
- persistence: defines whether AFT entries persist after a client disconnection

The first client to connect to the gRIBI server can set the redundancy and persistence parameters for all subsequent client connections. After the settings are defined, they cannot be altered until all clients have disconnected and all programmed entries are removed.

To define the session parameters, the client must send a ModifyRequest with the `session_parameters` field populated, but without any AFTOperations included.

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

Note: The client must send the `session_parameters` field before any AFTOperations. If a client sends a change to session parameters after a first-in AFTOperation, the gRIBI server ignores the change and returns an error.

The following table describes the available options for the sessions parameters.

Table 2: Session parameters options

Parameter	Options
redundancy	• **ALL_PRIMARY (Default)** The network device accepts AFTOperations from all clients. However, when a client adds an entry, only that same client can modify or delete the entry. When the device receives the ADD AFTOperation from any client, it adds an AFT entry to its gRIBI state. But the device only deletes an AFT entry when it receives a DELETE operation from the client that previously issued the ADD for that entry. Similarly, the device processes REPLACE AFTOperations only for AFT entries that the client previously added. • **SINGLE_PRIMARY** The network device accepts AFTOperations from the primary client only. In this case, the clients must take part in an election process (out of band of SR Linux) that identifies a single client as the primary client with the highest election ID.
persistence	• **DELETE (Default)** When a client disconnects, the network device deletes all AFT entries received from that client (first IPv4Entry, then NextHopGroup, then NextHop). • **PRESERVE** When the primary client disconnects (assuming SINGLE_PRIMARY redundancy), the network device preserves the routes programmed by the gRIBI server's RIB, the system RIB, and the system FIB. When a non primary client disconnects, no action is taken as all network element state is conveyed by the primary client. In this model, the gRIBI server builds state off of previously published information. This mechanism supports both a gRIBI server restart or a non-stop-forwarding failover to a standby CPM. Entries are not coupled to the liveliness of the Modify RPC. Entries persist until they are explicitly deleted, or the device performs a cold boot. **Note:** This is a deviation from the existing behavior expected in gRIBI.

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

3.1.2 Election ID

To handle scenarios where multiple clients consider themselves the primary client, each AFTOperation contains a new election ID. When the network device receives the election ID from the client, it compares the ID to the last known election ID. If the received election ID is less than the current election ID, it is ignored. If it is greater, the operation is accepted.

If the election ID is equal to the current ID, the client sending the update is accepted as the new primary client (in practice, this condition indicates a failure in the external primary election, but ensures a defined behavior for the device).

When the election ID in an AFTOperation does not match the current latest operation, an error is returned in the AFTResult indicating that the operation is not accepted.

Note: The gRPC session is not terminated in this case.

3.2 Get RPC

The Get RPC allows the client to retrieve the latest configuration from the network device. The client uses the Get RPC to retrieve the contents of the AFTs installed by the gRIBI server. The gRIBI server responds using a GetResponse stream containing the set of currently installed entries. When all entries are sent, the server closes the RPC.

The Get RPC is typically used for reconciliation between a client and a server after a disconnect or for periodic consistency checking.

The structure of the Get RPC is as follows:

```
rpc Get(GetRequest) returns (stream GetResponse);
```

3.2.1 Reconciliation

If a client becomes disconnected from the network device, it can reconcile the entries that are currently installed by the gRIBI server using the `Get` RPC.

The Get response returns the entries that the gRIBI server has installed on the device from any client. Only entries that have been acknowledged according to the defined per-session ACK requirement are returned to the gRIBI client; non-acknowledged entries are not sent. The client can use this information to identify the diff between the intended set of entries and the currently programmed set of entries and push the required updates.

An acknowledged entry is one for which the underlying ACK source (for example, hardware or RIB) has acknowledged the entry, instead of those for which an ACK was sent to a specific client. This addresses the case where a session failure occurs while an ACK is pending. Transmission of only acknowledged entries can result in a larger diff being sent by the client, but the intention of this mechanism is to ensure consistency at the end of the reconciliation process.

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

3.2.2 Application warm restart

Several events can occur on a device that can impact the liveliness of the gRIBI connection between the client and the network device. Particularly:

- Where redundant control-planes exist, control-plane failure can cause a reconnection of the gRIBI Modify RPC
- A gRIBI server restart on the device

In these cases, the device can determine the set of entries that were installed in gRIBI before the reconnection because these are programmed into the hardware persistently. After reconnection, the client can reconnect and perform a reconciliation.

After a warm restart, the gRIBI server behavior is as follows:

- All clients are disconnected and need to start new sessions
- Unreferenced next-hop entries are cleared (even if they were previously acknowledged in RIB_ACK mode)
- If an interface reference (interface_ref) was provided for a next-hop entry, it is not preserved after the restart
- If metadata was provided for an IP entry, it is not preserved after the restart
- If a next-hop within a next-hop-group has no weight, the weight value is set to 1 after the restart
- The highest known election ID is reset to zero after the restart
- Session parameters are cleared; the first client that connects after a warm restart can set the session parameters for all future clients

3.3 Flush RPC

The Flush RPC allows the client to remove all gRIBI routes from a specified network instance. This provides the client with a means to disable all gRIBI traffic engineering in case of system failure.

The client can send FlushRequests for specific network instances, and the `network_instance` is a mandatory field. The FlushRequest removes all installed AFT entries (IPv4Entry, NextHopGroup, and NextHop) in the specified network instance.

The FlushRequest also supports the option to flush all entries in all network instances.

The structure of the Flush RPC is as follows:

```
rpc Flush(FlushRequest) returns (FlushResponse);
```

3.4 Authentication

If the gRIBI server is not using UNIX sockets, sessions between the gRIBI client and SR Linux device must be encrypted using Transport Layer Security (TLS). Fall back to unencrypted sessions is not supported. You can specify TLS settings within a TLS profile and apply the TLS profile when configuring a gRIBI

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

server within a network instance. When the gRIBI server is enabled, gRIBI clients connect and authenticate to the SR Linux device using the settings specified in the TLS profile.

With **authenticate-client** set to **true** in the TLS profile, new connections are mutually authenticated. Each entity validates the X.509 certificate of the remote entity to ensure that the remote entity is both known and authorized to connect to the local system.

See the "Using SPIFFE for client authentication (mTLS)" section in the *SR Linux Configuration Basics Guide* for information about using SPIFFE for client authentication in TLS sessions.

For details about setting up a TLS profile, see the *SR Linux Configuration Basics Guide*

3.5 Event Logging

You can configure logging of gRIBI information (username, operation) to syslog. Some RPCs produce a volume of logging information that can overwhelm the syslog, so you can disable or abbreviate logging on a per-RPC basis. For more information about configuring syslog, see the Logging chapter of the *SR Linux Configuration Basics Guide* guide.

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

4 gRIBI configuration

To enable the gRIBI service:

1. Configure gRPC server settings at system level using the following procedures:
   a. Enabling gRIBI service support
   b. (Optional) Configuring a UNIX socket for the gRPC server
2. (Optional) Configure network instance level gRIBI protocol parameters:
   - Configuring gRIBI protocol parameters for the network instance

4.1 Enabling gRIBI service support

About this task

To configure the gRIBI service, you must enable gRPC server support.

Procedure

Step 1. Set the gRPC server **admin-state** to **enable** (default is **disable**):

system grpc-server <*name*> **admin-state enable**

Step 2. Set the following system-level gRPC server parameters:

- **metadata-authentication**: enables or disables the use of username/password authentication for every gRPC RPC request
- **network-instance**: specifies the network instance in which the gRPC listens for incoming connections.
- **port**: sets the TCP port the gRPC server listens on for incoming connections
- **Note:** 57400 is the default gRPC server port value, Any other value requires that you configure an ACL CPM filter to accept traffic on the configured port value.
- **rate-limit**: sets a limit on the number of connection attempts per minute
- **services**: specifies the gRPC service (in this case **gribi**) to enable on this gRPC server instance
- **session-limit**: sets a limit on the number of simultaneous active gRPC sessions
- **source-address**: lists the IP addresses the gRPC server listens on within the network instance
- **timeout**: sets the idle timeout in seconds on gRPC clients
- **trace-options**: sets gRPC trace options (**[common | grpc | request | response]**)
- **tls-profile**: references the TLS profile to use on the gRPC server (mandatory)
- **unix-socket**: enables or disables use of UNIX sockets for the gRPC server

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

- **yang-models**: specifies the YANG models to use when the origin field is not present in requests (default: **native**)

Example: Enable the gRIBI service support

The following example enables the gRIBI service on the mgmt gRPC server by specifying **gribi** as a supported service. It also sets the **timeout**, **rate-limit**, and **session-limit** values to their default values and sets the **trace-options** to **common**. It also defines a network-instance, source IP address, and port for the gRPC server.

```
--{ * candidate shared default }--[  ]--
# info with-context system grpc-server mgmt
    system {
        grpc-server mgmt {
            admin-state enable
            timeout 7200
            rate-limit 60
            session-limit 20
            network-instance mgmt
            port 57400
            trace-options [
                common
            ]
            services [
                gribi
            ]
            source-address [
                10.10.10.1
            ]
        }
    }
```

4.2 Configuring a UNIX socket for the gRPC server

About this task

To configure a UNIX socket for the gRPC server:

Procedure

Step 1. Set the system gRPC server UNIX socket **admin-state** to **enable** (default is **disable**):

system grpc-server unix-socket admin-state enable

Step 2. Set the following gRPC server parameters:

- **tls-profile**: References the TLS profile to use on the gRPC UNIX socket server. If none is specified, then TLS is not used.
- **metadata-authentication**: Enables or disables the use of username/password authentication for every gRPC RPC request (default: **true**).

Example

The following example enables a UNIX socket using a TLS profile (**test-tls**) and sets **metadata-authentication** to **true**.

```
--{ * candidate shared default }--[  ]--
# info with-context system grpc-server mgmt
```


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

```
    system {
        grpc-server mgmt {
            admin-state enable
            metadata-authentication true
            tls-profile test-tls
            unix-socket {
                admin-state enable
            }
        }
    }
```

4.3 Configuring gRIBI protocol parameters for the network instance

About this task

At the network instance level, you can configure gRIBI protocol parameters. By default, the gRIBI protocol **admin-state** is set to **enable** for the default network instance and for all IP-VRF network instances. However, state and telemetry information are not available for the **network-instance protocols gribi container** until a configuration exists in that context. Which means that you cannot display the **admin-state** for these network instances using **info from state** until you commit some configuration in the **network-instance protocols gribi** context. Nonetheless, these network instances remain available for gRIBI programming by default.

When the gRIBI protocol **admin-state** is set to **disable**, all IP entries and next-hop-groups associated with the network instance are deleted from the gRIBI service database. The recovery of this state depends on the gRIBI clients to re-signal all of the deleted entries. While in a disabled state, no entries are accepted for this network instance. (This is the same behavior when the network instance does not exist at all.)

Use the following procedure to configure gRIBI protocol parameters for the network instance.

Procedure

Step 1. To configure the gRIBI protocol **admin-state**, use the following command (default is **enable**):

network-instance *<instance>* **protocols gribi admin-state [enable | disable]**

Step 2. To set the gRIBI protocol parameters for the network instance, use the following parameters:

- **default-metric**: Sets the route table metric to use for all gRIBI-created IPv4 and IPv6 routes.
- **default-preference**: Sets the default preference when deciding the route to use from different protocols (lower values indicate higher preference).
- **max-ecmp-hash-buckets-per-next-hop-group**: Sets the maximum number of ECMP hash buckets per next-hop-group.
- **maximum-routes**: Sets the maximum number of gRIBI routes (sum of IPv4 and IPv6 entries).

Example

The following example sets the **admin-state**, **default-metric**, **default-preference**, **max-ecmp-hash-buckets-per-next-hop-group**, and **maximum-routes** parameters to their default values.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols gribi
    network-instance default {
        protocols {
            gribi {
                admin-state enable
```


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

```
                default-metric 1
                default-preference 6
                max-ecmp-hash-buckets-per-next-hop-group 128
                maximum-routes 0
            }
        }
    }
```


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

5 gRIBI display commands

SR Linux supports display commands to provide operational information about gRIBI clients and routes.

5.1 Displaying gRIBI client information

About this task

You can use the **info from state** command to display gRIBI client information.

Procedure

Enter the following command:

info from state system grpc-server <*name*> **client**<*id*>

Where <*id*> refers to the system generated ID for the client.

Available options are as follows:

- **election-id**: Election ID of this client
- **gribi persistence-mode**: The defined persistence mode as signaled by the client
- **remote-host**: Remote host of the client
- **remote-port**: Remote port of the client
- **rpc**: The called package, service, and RPC
- **start-time**: Time the client first connected
- **user**: Authenticated username for the client
- **user-agent**: User agent used for by the client

Example: info from state system grpc-server mgmt client election-id

```
--{ * candidate shared default }--[ ]--
# info from state with-context system grpc-server mgmt client 0 election-id
    system {
        grpc-server mgmt {
            client 0 {
                election-id 1:1
            }
        }
    }
```

Example: info from state system grpc-server mgmt client persistence-mode

```
--{ * candidate shared default }--[ ]--
# info from state with-context system grpc-server mgmt client 0 persistence-mode
    system {
        grpc-server mgmt {
            client 0 {
                persistence-mode preserve
            }
```


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

```
        }
    }
```

Example: info from state system grpc-server mgmt client remote-host

```
--{ * candidate shared default }--[ ]--
# info from state with-context system grpc-server mgmt client 0 remote-host
    system {
        grpc-server mgmt {
            client 0 {
                remote-host 10.10.0.1
            }
        }
    }
```

Example: info from state system grpc-server mgmt client remote-port

```
--{ * candidate shared default }--[ ]--
# info from state with-context system grpc-server mgmt client 0 remote-port
    system {
        grpc-server mgmt {
            client 0 {
                remote-port 33214
            }
        }
    }
```

Example: info from state system grpc-server mgmt client start-time

```
--{ * candidate shared default }--[ ]--
# info from state with-context system grpc-server mgmt client 0 start-time
    system {
        grpc-server mgmt {
            client 0 {
                start-time "31 seconds ago"
            }
        }
    }
```

Example: info from state system grpc-server mgmt client user

```
--{ * candidate shared default }--[ ]--
# info from state with-context system grpc-server mgmt client 0 user
    system {
        grpc-server mgmt {
            client 0 {
                user __gribi__
            }
        }
    }
```

Example: info from state system grpc-server mgmt client user-agent

```
--{ * candidate shared default }--[ ]--
# info from state with-context system grpc-server mgmt client 0 user-agent
    system {
        grpc-server mgmt {
            client 0 {
                user-agent "grpc-python/1.35.0 grpc-c/14.0.0 (linux; chttp2)"
```


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

```
            }
        }
    }
```

5.2 Displaying gRIBI route information

About this task

You can use the **show network-instance route-table** command to display gRIBI routes.

Procedure

Enter the following command:

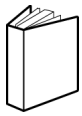
show network-instance <*id*> **route-table**

Example: show network-instance route-table

```
--{ * candidate shared default }--[ ]--
# show network-instance vrf route-table
-----------------------------------------------------------------------------------------------
IPv4 unicast route table of network instance vrf
-----------------------------------------------------------------------------------------------
+--------------+------+-----------+---------------+--------+---------+--------+--------+--------+
|   Prefix     |  ID  |   Route   |  Route Owner  | Active | Metric  |  Pref  | Next-  | Next-  |
|              |      |   Type    |               |        |         |        |  hop   | hop In |
|              |      |           |               |        |         |        | (Type) | terfac |
|              |      |           |               |        |         |        |        |   e    |
+==============+======+===========+===============+========+=========+========+========+========+
| 10.10.1.0/24 | 1    | gribi     | grpc_server   | True   | 1       | 4      |        |        |
+--------------+------+-----------+---------------+--------+---------+--------+--------+--------+
-----------------------------------------------------------------------------------------------
IPv4 routes total                    : 1
IPv4 prefixes with active routes     : 1
IPv4 prefixes with active ECMP routes: 0
-----------------------------------------------------------------------------------------------
```


© 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