Network Interface Shell

Release 23.7.R1

NISH Installation and User Guide

3HE 19230 AAAB TQZZA 01
Edition 01
July 2023

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

Table of contents

1 Getting started

1.1 About this guide

This guide describes how to install and use the Network Interface Shell (NISH), which includes the NISH client and manager service.

NISH is a solution for Linux platforms that provides users with a flexible way to manage multiple SR OS nodes running different SR OS versions, from the same Model-Driven Command Line Interface (MD-CLI) shell.

Note: This guide provides configuration examples based on MD-CLI syntax.

Topics described in this guide include:

- NISH solution overview
- NISH client installation and usage
- NISH manager service installation and usage

The following products currently support the use of the NISH features and commands described in this guide:

- 7750 Service Router
- Virtualized Service Router

Note:

- NISH requires a valid SR OS license for any compatible product.
- Configuration outputs shown in this guide are examples only and actual displays may differ depending on the user configuration.
- This guide covers content for the release specified on the title page of the guide, and may also contain content to be released in later maintenance loads. See the applicable software release notes for information about features supported in each load of the release software.

1.2 Audience

This guide is intended for network administrators who are responsible for installing, provisioning, and using NISH for SR OS node management via MD-CLI.

It is assumed that the network administrators have an understanding of the following topics:

- x86 hardware architecture
- Linux system installation, configuration, and administration methods
- networking principles and configurations

1.3 Related technical publications

After the installation process is completed, see the guides listed on the *7450 ESS, 7750 SR, and 7950 XRS Guide to Documentation* for information about the software configuration and CLI commands to use to configure the SR OS features and services.

1.4 Conventions

This section describes the general conventions used in this guide.

1.4.1 Precautionary and information messages

The following information symbols are used in the documentation.

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

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

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

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

1.4.2 Options or substeps in procedures and sequential workflows

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

Example: Options in a procedure

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

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

Example: Substeps in a procedure

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

2 Network Interface Shell

The Network Interface Shell (NISH) solution for Linux platforms provides remote control of SR OS MD-CLI and supports management of multiple SR OS nodes running different SR OS versions from the same MD-CLI shell. NISH provides similar functionality to operating MD-CLI directly on a node; however, the experience is not identical.

2.1 NISH overview

Modern network node architectures are moving toward disaggregated solutions for specific applications, such as the following examples:

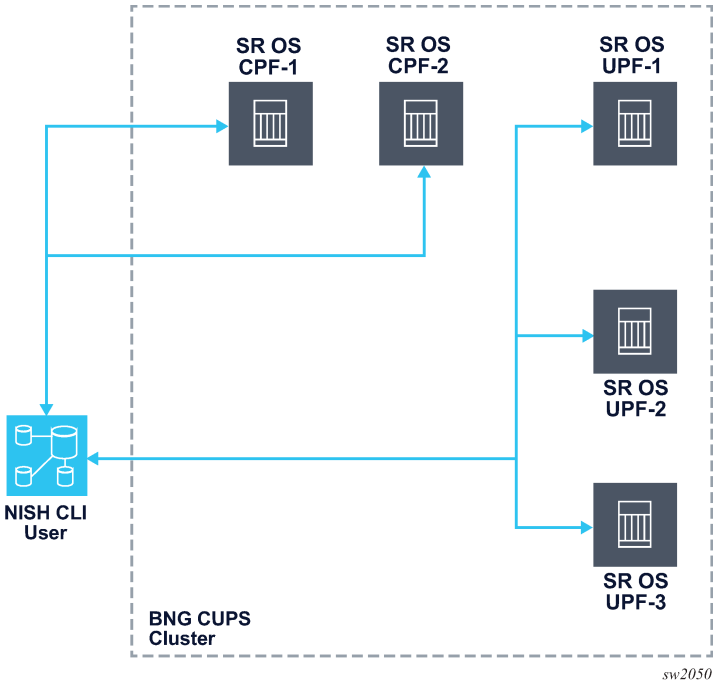
- Control and User Plane Separation (CUPS) deployments
- microservices operating as multiple, interconnected applications able to scale through independent addition and removal, run different software versions, and support rolling upgrades

The SR OS Broadband Network Gateway (BNG) CUPS feature is an example of a CUPS deployment architecture with multiple SR OS network nodes (both physical and virtual) operating as a cluster of independent nodes. By acting as one unit, the BNG CUPS cluster supports flexible and scalable BNG deployment.

For this type of CUPS deployment, the Network Interface Shell (NISH) solution for Linux platforms provides efficient remote control of the SR OS MD-CLI, by enabling management of multiple nodes that are running different versions of SR OS from the same MD-CLI shell.

Figure 1: NISH management of BNG CUPS cluster shows the NISH CLI management of a BNG CUPS deployment with two Control Plane function (CPF) nodes and three User Plane function (UPF) nodes.

Figure 1: NISH management of BNG CUPS cluster



2.2 NISH components

The NISH solution incorporates the following components:

- One or more network nodes (physical or virtual) running SR OS in MD mode is required, as well as an authorized user with gRPC access and the gRPC MD-CLI service enabled.
- The NISH client is required for the NISH solution. The NISH client is a utility that runs on a Linux platform and provides the interface to access SR OS nodes and use MD-CLI to configure a single node or multiple nodes defined either in a static local configuration file or dynamically using a NISH manager.
- The NISH manager is an optional service that runs on a Linux server. It listens for communications from SR OS nodes that are configured to use the SR OS remote management feature and creates a dynamic inventory of the nodes for one or more NISH clients. When the SR OS nodes register with the NISH manager, their profile information is provided during the connection.

  See the *7450 ESS, 7750 SR, 7950 XRS, and VSR System Management Guide*, section "Remote Management Using a Remote Network Interface Shell Manager ", for more information about the SR OS remote management feature.

2.3 Accessing the Linux man pages

About this task

Linux manual (man) pages are provided by the installation of the NISH client and the NISH manager. After you install NISH, you can access the man pages without having to launch the NISH client or manager.

Procedure

To access the Linux man pages for NISH, enter one of the following commands at the Linux terminal, depending on whether the NISH client or NISH manager is needed:

- For the NISH client, use **man nish**.
- For the NISH manager, use **man nish-manager**.

2.4 MD-CLI features support

The NISH solution for Linux platforms provides remote control of SR OS MD-CLI and supports management of multiple SR OS nodes running different SR OS versions from the same MD-CLI shell. The functionality that the NISH platform provides is similar to operating MD-CLI directly on a node; however, the experience is not identical.

The following are some of the MD-CLI features that NISH does not support:

- MD-CLI clear, monitor, show, and tools commands
- MD-CLI logging
- switching to the classic CLI
- XML and JSON output formatting
- BOF
- SSH
- Telnet

3 NISH client

This chapter provides information about the NISH client features and functionality, and procedures for installing and using NISH to manage multiple SR OS nodes using remote MD-CLI shells.

3.1 NISH client overview

This section describes some of the main features of the NISH client, including the interface for managing multiple SR OS nodes, navigation options, help and assistance options, direct access to Linux, SR OS node configuration, and operational state management.

3.1.1 SR OS node management

The NISH client provides an MD-CLI interface to multiple SR OS nodes. The NISH client uses the model-driven networking capabilities built into SR OS to identify the software version and obtain the MD-CLI schema for each SR OS node. This allows the NISH MD-CLI to generate the correct CLI tree that can be navigated within each SR OS node context. It also ensures that each node context is specific to its own software version.

The NISH client also allows the operator to issue Linux commands on the local machine, without leaving the MD-CLI shell. This capability provides increased productivity during extended management activities.

When launched, the NISH client provides an extensible MD-CLI interface. After the trademark, legal, and disclaimer information, NISH displays the following.

```
Starting nish application...
admin's password:
Connecting to test1 port 57400
!!! Creating unsecure connection !!!

Reading MD-CLI service capabilities...

Reading MD-CLI service event context...
[==============================================================================]

Reading MD-CLI service schemas...
[==============================================================================]

[]
A:admin@test1# ?

 admin                 + Enter the admin context
 configure             + Enter the configuration context
 environment           + Enter the environment configuration context
 file                  + Enter the file management context for file operations
 li                    + Enter the lawful intercept context
 password              - Change password command
 state                 + Show state information

 Global commands:
 back                  - Move back one or more levels
```

```
 delete                 - Delete an element from the candidate datastore
 edit-config            - Enter a candidate configuration mode
 enable                 - Enable administrative mode
 exec                   - Execute commands from a file
 exit                   - Return to the previous working context or to the →
operational root
 history                - Show the most recently entered commands
 logout                 - Exit the CLI session
 oam                    - Enter the oam context
 ping                   - Ping an IP address or DNS name
 pwc                    - Show the present working context
 ssh                    - Execute ssh command
 telnet                 - Telnet to IP address or DNS name
 top                    - Move to the top level of the context
 traceroute             - Trace route to a destination
 tree                   - Show the command tree under the present working context

[]
A:admin@test1#
```

3.1.2 SR OS life cycle support

Beginning in Release 21.5.R1, the NISH client supports up to two prior SR OS major releases. This backwards compatibility is effective from SR OS Release 20.10.R1. As a hypothetical example, the NISH client instance with a version number 23.5.R1 supports SR OS Release 21.2.R1 and later.

The NISH client ensures that the SR OS version of the device that it connects to conforms to the supported release interval. If the SR OS version is outside the supported range, the client fails to connect and provides an error message.

3.1.3 MD-CLI navigation in NISH client

Navigating in the NISH client MD-CLI interface is similar to navigating in the MD-CLI interface directly attached to a node. The NISH MD-CLI provides a YANG model-aware tree structure generated by the SR OS. Users can navigate the tree structure using the **tree**, **back**, **top**, and **exit** commands, enter a branch by typing the branch name into the CLI, and enter a list by typing the list name and key.

See the *7450 ESS, 7750 SR, 7950 XRS, and VSR MD-CLI User Guide*, for more information about navigating in MD-CLI.

3.1.4 NISH help and assistance

Context-sensitive help and auto-completion are supported within the NISH MD-CLI.

The Linux man pages provide usage assistance for NISH; see Accessing the Linux man pages for more information about how to access Linux man pages.

3.1.5 Linux direct access using NISH MD-CLI hybrid shell

Unlike operating directly in MD-CLI, the NISH MD-CLI provides a hybrid shell that allows users to execute Linux commands directly on the local workstation, without leaving the MD-CLI shell. This provides a flexible working environment for managing multiple nodes.

The following example shows output of a **configure system** setup, with execution of both Linux and MD-CLI commands.

```
(pr)[configure system]
A:admin@test1# ls -la
total 96
dr-xr-x---.  7 root root  4096 Jun 11 13:06 .
dr-xr-xr-x. 18 root root   261 Apr 15 13:21 ..
-rw-------.  1 root root  1860 Apr 15 12:00 anaconda-ks.cfg
-rw-------   1 root root 21113 Jun 11 13:06 .bash_history
-rw-r--r--.  1 root root    18 May 11  2019 .bash_logout
-rw-r--r--.  1 root root   176 May 11  2019 .bash_profile
-rw-r--r--   1 root root   290 May 11 18:24 .bashrc
drwx------   4 root root    32 May 22 12:30 .cache
drwx------   3 root root    18 Apr 15 15:44 .config
-rw-r--r--   1 root root     0 May 22 11:04 config
-rw-r--r--.  1 root root   100 May 11  2019 .cshrc
drwxr-xr-x   2 root root    25 May  6 17:23 .docker
-rw-r--r--   1 root root    62 May  6 13:06 .gitconfig
-rw-------   1 root root    97 Jun  1 14:58 .lesshst
lrwxrwxrwx   1 root root    23 Apr 15 13:28 misc_vol -> /media/gluster/misc_vol
-rw-------   1 root root     0 Jun  4 12:16 .python_history
-rw-r--r--   1 root root 30123 May  6 16:14 screenlog.0
drwx------   2 root root    57 May  6 13:26 .ssh
-rw-r--r--.  1 root root   129 May 11  2019 .tcshrc
-rw-------   1 root root   533 Apr 15 14:50 .viminfo

(pr)[configure system]
A:admin@test1# cat .bashrc
# .bashrc

# User specific aliases and functions
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
. /etc/bashrc

fi
```

If a command exists in both the MD-CLI and the Linux shell, the MD-CLI command takes precedence. This is shown in the following output example, which uses the **time** command.

```
(pr)[configure system]
A:admin@test1# time echo “hello world”
                    ^^^^
MINOR: MGMT_CORE #2201: Unknown element - 'echo'
(pr)[configure system]
A:admin@test1# time
(pr)[configure system time]
A:admin@test1# time echo “hello world”
"hello world"
0.00user 0.00system 0:00.00elapsed 93%CPU (0avgtext+0avgdata 1944maxresident)k
0inputs+0outputs (0major+84minor)pagefaults 0swaps

(pr)[configure system time]
A:admin@test1#
```

3.1.6 SR OS node configuration using NISH MD-CLI

The NISH MD-CLI provides the ability to see and edit individual router configurations using the transactional configuration modes available in MD-CLI (private, exclusive, global, and read-only).

The following example shows this ability.

```
[]
A:admin@test1# edit-config private
INFO: CLI #2070: Entering private configuration mode
INFO: CLI #2061: Uncommitted changes are discarded on configuration mode exit

(pr)[]
A:admin@test1# configure system

(pr)[configure system]
A:admin@test1# name example

*(pr)[configure system]
A:admin@test1# compare
-   name "test1"
+   name "example"

*(pr)[configure system]
A:admin@test1# commit

(pr)[configure system]
A:admin@example#
```

3.1.7 Obtaining the SR OS node operational state using NISH MD-CLI

The NISH MD-CLI provides the ability to obtain the operational state of the SR OS node, using the **info** command within the `state` tree, as shown in the following example.

```
[]
A:admin@example# /state

[state]
A:admin@example# system

[state system]
A:admin@example# info | match oper-name
    oper-name "example"
```

3.2 NISH client installation

The NISH client is delivered as a set of RPM packages and is supported on the CentOS 7 Linux distribution.

The NISH RPMs are placed into a local directory on the target machine and installed using the YUM package manager built into CentOS 7.

3.2.1 Installation process

The NISH installation process performs the following activities:

- installs the NISH client into `/usr/bin/nish`
- installs the End User License Agreement (EULA) in `/usr/share/licenses/nish/EULA`
- installs an example connections file in `/etc/nish/connections`
- installs an example local schema file in `/etc/nish/local-schema`
- installs the global NISH run commands (`rc`) file into `/etc/nish/nishrc`
- installs the Linux man pages for the **nish** command

See Connections file, Local schema file, and NISH rc files, respectively, for more information about connections, local schema files, NISH `rc` files, and the features they support.

See Installing the NISH client for the procedure to install the NISH client.

3.2.2 Installing the NISH client

Prerequisites

Before performing the installation, ensure that you have the following:

- RPMs downloaded from the Nokia support portal
- CentOS 7
- Linux administrative user access rights

To install the NISH client, perform the following steps:

Procedure

Step 1. Put the NISH client RPM packages in a local directory on the target machine.

Step 2. Use the YUM package manager built into CentOS 7 to install the NISH RPMs.

Example

```
[root@server ~]# yum localinstall -y nish-23.3-R1.el7.x86_64.rpm
# Last metadata expiration check: 2:33:56 ago on Sun 1 Jan 2023 11:42:12 CEST.
Dependencies resolved.
================================================================================
Package     Architecture    Version       Repository            Size
================================================================================
Installing:
 nish       x86_64          23.3-R1.el7   @commandline          1.4 M
Transaction Summary
================================================================================
Install  2 Packages
Total size: 5.3 M
Installed size: 18 M
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
1/1  Installing       : nish-23.3-R1.el7.x86_64
```

```
1/1  Running scriptlet: nish-23.3-R1.el7.x86_64
1/1  Verifying        : nish-23.3-R1.el7.x86_64
Complete!
```

3.3 NISH client operation modes

When installed on the Linux platform, the NISH client offers the following three modes of operation:

- **1:1 direct mode**

  In 1:1 direct mode, the NISH client connects directly to a single SR OS node and operates the MD-CLI from a local Linux workstation; see Using the NISH client in 1:1 direct mode.

- **1:n static mode**

  In 1:n static mode, the NISH client manages multiple SR OS nodes from a single MD-CLI shell, operates the MD-CLI from a local Linux workstation, and defines the SR nodes statically in a local configuration file; see Using the NISH client in 1:n static mode.

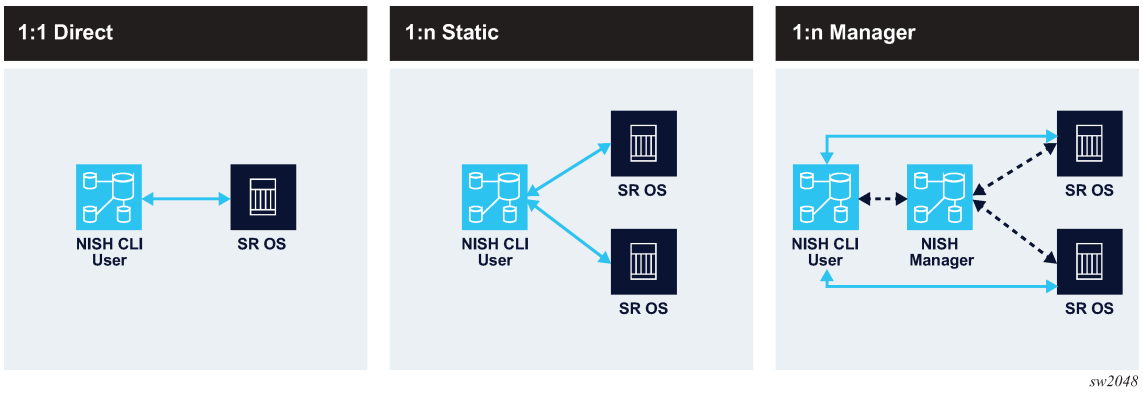
- **1:n manager mode**

  In 1:n manager mode, the NISH client manages multiple SR OS nodes from a single MD-CLI shell, operates the MD-CLI from a local Linux workstation, uses a Linux service to dynamically identify the SR OS nodes that are available, and automatically tracks additions and removals from the SR OS node clusters; see Using the NISH client in 1:n manager mode.

Note: The 1:n manager mode requires the installation of the NISH Manager service; see NISH manager.

Figure 2: NISH client operation modes shows the node configurations for the modes of operation supported by the NISH client.

Figure 2: NISH client operation modes

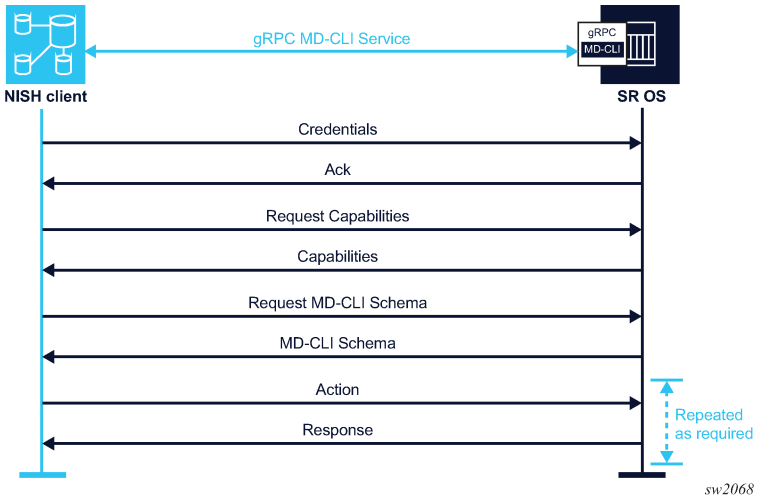


3.3.1 Using the NISH client in 1:1 direct mode

In 1:1 direct mode, the NISH client connects directly to a single SR OS node and operates the MD-CLI from a local Linux workstation.

The following figure shows the communication for 1:1 direct mode.

Figure 3: Communication between the NISH client and the SR OS node



Prerequisites

Review the following requirements before using the NISH client in 1:1 direct mode:

- Ensure that you have permission to access the gRPC interface and the MD-CLI gRPC service of the target node.
- The **nish** command can be issued from the command line, providing a username, hostname (or IP address), and port number of the target SR OS node. Alternatively, the hostname and the port number can be defined using environment variables or in a NISH `rc` file. In this case, the **nish** command can be entered without these arguments. See NISH rc files for more information about configuring `rc` files.

Note: The arguments defined in environment variables or in the NISH `rc` file must be supported in the current mode of operation. If arguments for another mode of operation are defined, the NISH client fails and returns an error.

To use the NISH client in 1:1 direct mode, perform the following steps:

Procedure

Step 1. Execute the **nish** command from the command line, providing a username, hostname (or IP address), and port number of the target SR OS node:

nish [*OPTIONS*] [**-l** *user*] [**-p** *port-number*] [*user*]@*hostname*

You can specify the user with the **-l** flag or **--login-name** option, together with the hostname in the format user@hostname, or in the $NISH_USER environment variable. If you do not specify a user with any of these options, the default $USER environment variable is used.

The port number defines the target node TCP port. If it is not set, the default port 57400 is used. To specify a non-default port number use the **-p** flag or **--port** option.

The hostname defines the target SR OS node. FQDN, hostname, IPv4 address, or IPv6 address formats are accepted. See Command options, for information about other command options.

The following is an example of **nish** command usage.

```
[root@server ~]# nish admin@test1
```

In this example:

- The name of the SR OS node user who has the required permissions is admin.
- The hostname of the node is test1.
- 57400 is the default port and does not need to be specified explicitly.

Note: If the user, target host, and target port are defined in environment variables or in a NISH `rc` file, and no other options are needed, the **nish** command can be used without these arguments.

Step 2. Enter the password for the SR OS node when prompted.

3.3.2 Using the NISH client in 1:n static mode

Prerequisites

Review the following requirements before using the NISH client in 1:n static mode:

- A `connections` file is required. The `connections` file contains the details of the SR OS nodes that are available for direct connections. The `connections` file may list multiple SR OS nodes. For more information, see Connections file.

  Note: The presence of a node in the `connections` file does not guarantee that the referenced node is online and available to receive NISH connections.

- The referenced node must be configured correctly with gRPC, a user with gRPC access, and the gRPC MD-CLI service.
- The correct TLS-enabled connectivity is recommended.

  See the 7450 ESS, 7750 SR, 7950 XRS, and VSR documentation for more information.

To use the NISH client in 1:n static mode, perform the following steps:

Procedure

Step 1. Execute the **nish** command from the command line, providing the relative or absolute path to the `connections` file.

nish [*OPTIONS*] [**-s** *local-schema-file*] **-c** *connection-file*

You can specify the absolute or relative path to the `connections` file with the **-c** flag or **--connection-file**, in a NISH `rc` file or in an environment variable; see NISH rc files. In this case, the **nish** command can be entered without this argument. Arguments defined in the NISH `rc` file or using the environment variable must be supported in the current mode of operation. If an argument for another mode of operation is defined, the NISH client fails and returns an error.

Note: If the path to the `connections` file is missing or incorrect, the NISH application starts without available nodes. The output of the command shows that the `connections` file is missing.

The `local schema` file contains the details of the locally defined NISH MD-CLI schema; see Local schema file. For information about the other options, see Command options.

In the following example, the locally defined `schema` file is absent and the `connections` file is present in the same directory as the **nish** command.

```
[root@server ~]# nish -c connections-demo
```

When the path to the `connections` file is present and correct and no local `schema` file is defined, NISH starts with a minimal built-in MD-CLI schema. This schema defines a number of standard commands and some MD-CLI branches. The following output shows an example.

```
Starting nish application...
Reading local schema...
[==============================================================================]
[]
root@#
 connections           + Configuration of all devices.
 environment           + Enter the environment configuration context
 Global commands:
 back                  - Move back one or more levels
 delete                - Delete an element from the candidate datastore
 exec                  - Execute commands from a file
 exit                  - Return to the previous working context or to →
the operational root
 history               - Show the most recently entered commands
 logout                - Exit the CLI session
 pwc                   - Show the present working context
 top                   - Move to the top level of the context
 tree                  - Show the command tree under the present working context
```

Step 2. Navigate to the `connections` branch to access the SR OS node (where each node is shown under the `connection` list) by entering the **connection** command followed by the name of the node.

When navigating to the SR OS node in the tree, NISH prompts for the username and password of an SR OS user with the appropriate access and permissions.

Step 3. Enter your SR OS username and password with the appropriate access and permissions.

Example

```
[connections]
root@# connection <tab>
 test1
 test2

[connections]
root@# connection test1
Login: admin
admin's password: password

Connecting to 172.16.123.1 port 57400
!!! Creating unsecure connection !!!

Reading MD-CLI service capabilities...
WARNING: Unknown capability
Reading MD-CLI service event context...
[==============================================================================]

Reading MD-CLI service schemas...
[==============================================================================]

[connections connection "test1"]
```

```
A:admin@test1#
```

The connection list name is a device label. The **connection** device label is a reserved label and displays all routers that can be managed by NISH, regardless of any more granular groupings. For more information about device labels, see Device labels.

Navigation to alternative SR OS nodes within the MD-CLI interface is supported using the usual navigation techniques. When entering a new node, NISH prompts for the user name and password for that node. The credentials for each node are cached during the established session between the NISH client and the SR OS node. See Authentication credentials for more information about credentials.

Step 4. Use the usual MD-CLI navigation techniques to navigate to alternative SR OS nodes within the MD-CLI interface.

The following example shows a multinode navigation.

Example

```
Starting nish application...
Reading local schema...
[==============================================================================]
[]
root@# connections connection ?

 test1
 test2

[]
root@# connections connection test1
Login: admin
admin's password: password
Connecting to 172.16.123.1 port 57400
!!! Creating unsecure connection !!!

Reading MD-CLI service capabilities...
Reading MD-CLI service event context...
[==============================================================================]
Reading MD-CLI service schemas...
[==============================================================================]

[connections connection "test1"]
A:admin@test1# back
[]
root@# connections connection test2

Login: admin
admin's password: password
Connecting to 172.16.123.2 port 57400
!!! Creating unsecure connection !!!

Reading MD-CLI service capabilities...
Reading MD-CLI service event context...
[==============================================================================]
Reading MD-CLI service schemas...
[==============================================================================]

[connections connection "test2"]
A:admin@test2# back

[]
root@# connections connection test1
```

```
[connections connection "test1"]
A:admin@test1#
```

3.3.3 Using the NISH client in 1:n manager mode

Prerequisites

Review the following requirements before using the NISH client in 1:n manager mode:

- A working NISH manager with known IP address and port number is required; see NISH manager.
- If the NISH manager is on a different host from the NISH client, the firewall permissions between the NISH client and the NISH manager must be set correctly.

To use the NISH client in 1:n manager mode, perform the following steps:

Procedure

Step 1. Execute the **nish** command from the command line, providing the IP address and port number to which the NISH manager service is bound:

nish [*OPTIONS*] [**-s** *local-schema-file*] **-m** *address:port*

You can specify the IP address and port number of the NISH manager with the **--manager** option or **-m** flag, in a NISH `rc` file or in an environment variable; see NISH rc files.

Note: If the IP address and port number of the NISH manager are defined in environment variables or in a NISH `rc` file, you can enter the **nish** command without these arguments. The arguments defined using environment variables or in the NISH `rc` file must be supported in the current mode of operation. If arguments for another mode of operation are defined, the NISH client fails and returns an error.

The local `schema` file contains the details of the locally defined NISH MD-CLI schema; see Local schema file. For information about the other options, see Command options.

In the following example, the local `schema` file is absent and the NISH manager service is bound to IP address 127.0.0.1 (the localhost) and port number 57400.

```
[root@server ~]# nish -m 127.0.0.1:57400
```

Step 2. Navigate and operate in NISH manager mode using the same methods used in NISH static mode.

The navigation and operation of NISH in manager mode is the same as in static mode. The NISH client makes direct connections to the nodes and traverses into them within the NISH MD-CLI; see Using the NISH client in 1:n static mode.

When the NISH client starts in manager mode, it makes a gRPC connection to the NISH manager service and obtains the inventory of nodes. The NISH manager uses an ON_CHANGE gRPC streaming method to ensure the inventory of nodes is continually updated.

3.4 Command options

The Linux man pages provide an overview of the options and arguments for the **nish** command. The following table describes some of the options and flags that can be used when starting the NISH client.

*Table 1: Options and flags for the **nish** command*

Option or flag	Description	References
--credential-cache	Enables caching of SR OS node credentials in static or manager mode	See Authentication credentials
--ca-cert **-t**	Specifies TLS mode	See Configuring NISH security
--color **NISH_COLORS**	Specifies the color scheme for the NISH environment	See NISH rc files
--no-proxy	Disables system-defined proxy settings and ignores the http_proxy and https_proxy environment variables	—
--disable-auto-completion-authorization	Disables authorization checks for the **auto-completion** command [1]	—

3.5 Authentication credentials

By default, the NISH client prompts for user credentials when connecting to each SR OS node. The user credentials are cached for the duration of the NISH client session with the node.

It is common in SR OS node cluster deployments for a user's credentials to be the same over all devices. This is either because a centralized provisioning and management system configures the user manually on all devices, or because an external system such as TACACS or RADIUS manages the user.

The NISH client can streamline such environments by caching a single username and password combination and automatically authenticating against each node as the user navigates into them, without further prompting.

To enable this feature, start the NISH client with the **--credential-cache** option.

When the NISH client starts with this feature enabled, it prompts immediately for the username and password combination, and caches this information until it closes.

When the credentials are cached and a user connects to a specific node, the NISH client attempts to authenticate using the global cached password. If this password is incorrect, the user is prompted to re-enter the password. If the authentication is successful, the password is cached for the duration of the NISH client session for that node only. This process is repeated each time the user connects to another node.

[1] Auto-completion ensures that only authorized commands are displayed in the auto-completion. The user's authorization to execute a command is checked against an authorization server or local profile. Disabling this option can improve performance but allows the auto-completion to display all commands, including the commands that cannot be executed because they fail authorization. Users are not able to execute commands for which they do not have authorization. See the *7450 ESS, 7750 SR, 7950 XRS, and VSR System Management Guide*, section "Authorization" for more information.

Credential caching can also be enabled in the NISH `rc` file; see Table 2: NISH rc file variables. If credential caching is enabled within the NISH `rc` file but is not needed for a specific session, you can disable it using the **--no-credential-cache** option.

Note: Both the NISH client static and manager modes support caching of credentials.

3.6 Device labels

Using device labels, an operator can group SR OS nodes within the NISH MD-CLI schema. A device label is a string value chosen by the operator to reference a group of SR OS nodes.

The **connection** device label is present by default and contains all SR OS nodes.

Groups are not predefined. The operator has the freedom to group nodes in the best way for the organization. The following are examples of SR OS node groupings:

- In a BNG CUPS deployment, the operator groups the user plane nodes with the UPF device label and the control plane nodes with the CPF device label.
- In a multi-platform deployment, the operator groups the SR OS nodes running on physical 7750 SR routers with the Physical device label and the VSR nodes with the Virtual device label.

Note: Device labels can only contain letters and integers. Special characters are not supported.

The operator defines the device labels in a `local schema` file. The NISH schema in the NISH MD-CLI uses the device labels in the `local schema` file to group SR OS nodes; see Local schema file.

In the NISH client static mode, the operator applies the device labels defined in the `local schema` file to the nodes in the `connections` file; see Connections file.

In the NISH client manager mode, the operator applies the device labels defined in the `local schema` file to the nodes, using the SR OS remote management feature. For more information about the configuration of remote management on an SR OS device, see the *7450 ESS, 7750 SR, 7950 XRS, and VSR Classic CLI Command Reference Guide*, section "gRPC MD-CLI".

When the device labels are defined in the `local schema` file and applied to the SR OS nodes, the NISH MD-CLI shows the SR OS nodes in the branch that matches the device label, as shown in the following example:

- test1 is configured with the device label UPF.
- test2 is configured with the device label CPF.

```
[]
root@# connections <tab>
 connection  UPF  CPF
[]
root@# connections UPF <tab>
 <ip>
 test1
[]
root@# connections CPF <tab>
 <ip>
 test2
[]
root@# connections connection <tab>
```

```
<ip>
test1
test2
```

Note: A device can have only one user-defined device label and has by default the system-defined **connection** device label.
If the operator applies to an SR OS node a device label that is not defined in the `local schema` file, the node appears only under the **connection** device label. Only device labels defined in the `local schema` file are visible in the NISH MD-CLI.

If the operator defines a device label in the `local schema` file and does not apply it to a node, the device label appears in the NISH MD-CLI without any member nodes.

3.7 Connections file

A NISH client operating in static mode uses a `connections` file. The `connections` file is a text file that contains the details of the SR OS nodes that are available within the NISH MD-CLI.

There are no requirements for the filename.

The following are required for the `connection` file:

- The NISH client must have access to the file location.
- The Linux user running the NISH client must have read permissions for the file.

The `connections` file defines a registration for each SR OS node. Each registration has the following fields:

- **id**

  The **id** field is the identifier string, which is usually the system name.

- **label**

  The **label** field may contain an optional device label, which can be used to add the node to a group; see Device labels for more information.

- **address**

  The **address** field consists of the IP address and port number to which the gRPC server on the SR OS node is bound. If a proxy server is used to facilitate IP address or port redirection, the IP address or port number of the proxy server can be entered.

The following example shows a `connections` file with one registration.

```
registration {
  id: "routername"
  label: "myLabel"
  address {
    address: "10.11.12.13"
    port: 57400
  }
}
```

Note: String fields are enclosed in quotation marks.

Multiple devices are added in serial order within a `connections` file. The following example shows a `connections` file with two registrations.

```
registration {
  id: "test1"
  label: "UPF"
  address {
    address: "172.16.123.1"
    port: 57400
  }
}
registration {
  id: "test2"
  label: "CPF"
  address {
    address: "172.16.123.2"
    port: 57400
  }
}
```

After installation, there is an example `connections` file in `/etc/nish/connections`. The `connections` file can be edited at any time. The NISH client periodically reads the `connections` file and updates the list of nodes.

Note: If the `connections` file is updated while using the NISH client, the operator should disable auto-save.

3.8 Local schema file

When the NISH client starts, it provides an initial MD-CLI tree to the user. This MD-CLI environment is based on a local MD-CLI schema definition. The NISH client provides a default built-in local schema. To enable user extensibility in the MD-CLI environment, the operator can create a locally defined MD-CLI `schema` file.

The `local schema` file is a text file that contains the details of the locally defined NISH MD-CLI schema extensibility definitions. It can be used in static and manager mode. There are no requirements for the filename.

The following are the requirements for the `local schema` file:

- The NISH client requires access to the file location.
- The Linux user running the NISH client requires read permissions for the file.

Note: It is not mandatory for the local schema to be the same for all users of the NISH client in static or manager modes. This allows administrators and users to define customized MD-CLI schemas for their own purposes.

To use a `local schema` file, start the NISH client with the **--local-schema-file** option or **--s** flag followed by the relative or absolute path to the `local schema` file, or define the path to the `local schema` file in an environment variable or a NISH `rc` file. See NISH rc files.

The `local schema` file defines a structure called a schema that contains one or more schema branches that can be linked hierarchically. The `schema` branches are either local or remote:

- **local schema**

Local schemas define branches in the NISH MD-CLI tree that operate locally on the Linux host; these are currently not supported.

- **remote schema**

  Remote schemas define branches in the NISH MD-CLI tree that connect to SR OS nodes.

The schema names create hierarchical structures within the MD-CLI and define device labels. The **connection** and **all** device labels are reserved and cannot be used; see Device labels.

The following example shows a `local schema` file with a branch containing two remote schemas. The **devices** schema defines the devices branch within the NISH MD-CLI. The sub-schemas **UPF** and **CPF** are remote schemas under the devices branch.

```
schemas {
  schema {
    name: "devices"
    shortDescription: "Configuration of all devices"
    schema {
      name: "UPF"
      shortDescription: "User plane devices"
      schemaType: remote
    }
    schema {
      name: "CPF"
      shortDescription: "control plane function devices"
      schemaType: remote
    }
  }
}
```

Note: If reserved device labels are used in the `local schema` file, the NISH client exits with an error, as shown in the following example.

```
ERROR: Specified local-schema-file contains reserved schema name
 'connection'
```

After the initial installation, there is an example `local schema` file in `/etc/nish/local-schema`.

3.9 NISH rc files

NISH provides the ability to place commonly used attributes into one or more run command (`rc`) files. A NISH `rc` file defines variables used when starting the NISH client on the command line. When a NISH `rc` file is present, the user does not need to enter the defined variables manually on the command line.

After the NISH client installation, there is an example NISH `rc` file in `/etc/nish/nishrc`.

A NISH `rc` file can be located in the file system as follows:

- The file can be present in `/etc/nish/nishrc` where it has global significance. All user instances of the NISH client use the global file.
- The file can be present in `~/.nishrc` where it has local user significance (the tilde symbol (~) is equivalent to the user's home directory, or could also be referenced as `$HOME/.nishrc`). The specific user instance of the NISH client uses the local file. The local file takes precedence over the global file.

A user can also set the variables supported in the NISH `rc` file as Linux environment variables. The user-defined environment variables take precedence over the local and central NISH `rc` file.

User-specific command line arguments take precedence over all variables defined in the global file, the local file, or in user-defined environment variables.

Table 2: NISH rc file variables describes the supported variables in a NISH `rc` file.

Table 2: NISH rc file variables

Variable	Description	Notes
NISH_COLORS	Specifies the color scheme for the NISH environment, using the same syntax rules as the **grep** command	Options: • **mo** - mode • **co** - context • **uc** - uncommitted changes • **cs** - card slot • **un** - username • **at** - at (@) • **sn** - system name • **pc** - prompt character • **er** - error message • **if** - info message • **wa** - warning message • **nt** - normal text
NISH_CONNECTIONS_FILE	Defines the path to the `connections` file	See Connections file
NISH_CREDENTIAL_CACHE	Enables the credential cache	Boolean See Authentication credentials
NISH_HOST	Defines the address of the target node	—
NISH_LOCAL_SCHEMA_FILE	Defines the path to the `local schema` file	See Local schema file
NISH_MANAGER	Defines the IP address and port number of the NISH manager service in the format IP:PORT	See Using the NISH client in 1:n manager mode
NISH_PORT	Defines the port of the target node	—
NISH_SERVER_CACHE_PATH	Defines the location of the cached downloaded device schemas	—

Variable	Description	Notes
NISH_USER	Defines the username that used to connect with the target node	—

3.10 Multinode operations

The NISH client provides the ability to manage multiple nodes from a single MD-CLI shell.

When operating in 1:n manager or 1:n static mode, the NISH client generates a pseudo-node named "all" within the connection device-label (`/connections connection` "all" in the MD-CLI path). The operator can navigate to this pseudo-node in the same way as to a single node within the NISH client. The pseudo-node addresses supported commands to all nodes that the NISH client knows about.

In 1:n manager or 1:n static mode, the NISH client also generates a pseudo-node named "all" in each device-label context it is aware of. By navigating into the pseudo-node within a specific device-label, the supported commands are addressed to all the nodes with that specific device-label that the NISH client knows about.

3.10.1 Multinode configuration modes

The exclusive candidate configuration mode is the only supported candidate configuration mode for NISH client multinode operation.

To access the exclusive candidate configuration mode, use the explicit **edit-config exclusive** command or the implicit **configure exclusive** command.

When the operator tries to access the exclusive candidate configuration mode within a multinode context, the NISH client attempts the following actions:

1. connects to each SR OS node prompting for authentication credentials, if required
2. obtains an exclusive lock on all the nodes

A warning is displayed for every node for which the NISH client cannot obtain an exclusive lock. The operator can optionally continue the configuration activities with the remaining nodes or exit from configuration mode.

A list of the nodes for which the NISH client has successfully obtained an exclusive candidate configuration lock is stored in memory, and the MD-CLI configuration commands are executed against this set of nodes.

Note: The list of nodes is created to ensure that operators have full knowledge of the set of nodes they are addressing with the configuration changes.
Any node that is added, removed, renamed, or re-labeled using the NISH connections file or from the NISH manager while the exclusive lock remains in force, is ignored until the operator returns to the operational mode.

When the operator is presented with the exclusive configuration mode, the prompt changes. When the operator leaves the exclusive configuration mode, all candidate configurations on all nodes are discarded.

3.10.2 Multinode info operation

The NISH client can perform the **info** operation on multiple SR OS nodes that share a specific device-label grouping.

If the output is identical on all the SR OS nodes in the device-label group, the **info** operation displays a single copy of the output.

If any of the output differs, the NISH client displays the SR OS node output with a banner separating the output of each node.

3.10.3 Multinode combined schemas

The NISH client can execute many SR OS operations on multiple SR OS nodes that share a specific device-label group.

The NISH client obtains details of the MD-CLI schema from each node in the device-label group and builds a combined schema. The generated combined schema includes all the fields that are common between the SR OS devices, along with the most restrictive set of validation rules. The NISH client then uses this combined schema for functions such as configuration, info, and compare, that are performed for the device-label group.

Note: The NISH client supports device-label groups that consist of SR OS nodes with different platform types and software versions. The more diverse these factors are, the smaller the supported combined schema.

3.10.4 Multinode compare operation

The NISH client can perform a compare operation on multiple SR OS nodes that share a specific device-label group. The multinode compare operation uses the combined schema; see Multinode combined schemas for more information.

If the output is identical on all the SR OS nodes in the device-label group, the compare operation displays a single copy of the output. If any of the node outputs differ, they are displayed separately with a banner between the output of each node.

3.10.5 Multinode interactive configuration

The NISH client can interactively configure multiple SR OS nodes that share a specific device-label group. Interactive configuration of multiple SR OS nodes uses the combined schema. This means that only configuration options that can be applied to all nodes in the group are available to the operator. Context-sensitive help and auto-completion are supported using the combined schema. See Multinode combined schemas for more information.

When you configure multinode interactive mode, the NISH client immediately places entries into the exclusively locked candidate configuration of every SR OS node in the group. Exiting the interactive configuration mode of a device-label group releases the exclusive lock on all devices that are part of the group. In-progress configuration changes that are stored in the candidate configuration but not committed are lost.

Only one device-label group at a time can hold the configuration lock. If the operator attempts to obtain a configuration lock from another device-label group, the system issues a prompt to release the previous lock. When the system releases the lock, all uncommitted configuration changes are discarded.

3.10.6 Configuration load operation

The NISH client provides the ability to provision a template-based configuration on multiple SR OS nodes with a specific device-label grouping. This operation is completed using the MD-CLI **load** command.

When using the **load** command within NISH for multinode configuration, the **merge** and **full-replace** options are supported. The referenced configuration file supplied to the **load** command must be accessible from each individual SR OS node within the addressed set of nodes, unless it is prefixed with the URL prefix `nish://`. When `nish://` is used, the relative or absolute path or filename that follows `nish://` is treated as local to the NISH client.

Note: The MD-CLI **load** command is the only supported method for multinode configuration from the NISH client.
The **load** command within a multi-device context (MDC) is atomic, that is, it succeeds on all addressed SR OS nodes or no SR OS nodes.

See the *7450 ESS, 7750 SR, 7950 XRS, and VSR MD-CLI Command Reference Guide* for more information about the **load** command.

The following table describes examples of the supported file URL options that can be used to reference the configuration template file when using the **load merge** and **load full-replace** commands for multinode configuration from the NISH client. In the examples, a configuration file called `myconfig.cfg` is referenced.

Table 3: File URL reference options for NISH client load merge and replace

Command with file URL reference	Description
load merge `cf3:/myconfig.cfg`	The `myconfig.cfg` file containing the configuration changes must be accessible in the root directory of `cf3:`, on each node in the set of addressed nodes.
load merge `ftp://`*`user`*`:`*`password`*`@100.0.0.1/myconfig.cfg`	Each node in the set of addressed nodes must be able to successfully connect to the FTP server at 100.0.0.1 and authenticate using the user and password credentials.
load merge `nish://myconfig.cfg`	Each node in the set of addressed nodes is configured by merging the file `myconfig.cfg` located in the current working directory of the host operating system on which the NISH client is running.
load merge `nish:///`*`home/username`*`/myconfig.cfg`	Each node in the set of addressed nodes is configured by merging the `myconfig.cfg` file located in the home directory of the user on the host operating system on which the NISH client is running.

3.10.7 Multinode rollback operation

The NISH client supports rollback of configuration changes on multiple nodes at the same time, using the **rollback** command. Issuing the **rollback** command from a multinode context in the NISH client causes every node in the group to attempt to roll back to the specified checkpoint. If all the nodes successfully roll back to the specified checkpoint, the operation completes successfully. If any node in the group cannot roll back, the command fails and none of the nodes roll back.

3.10.8 Multinode commit operation

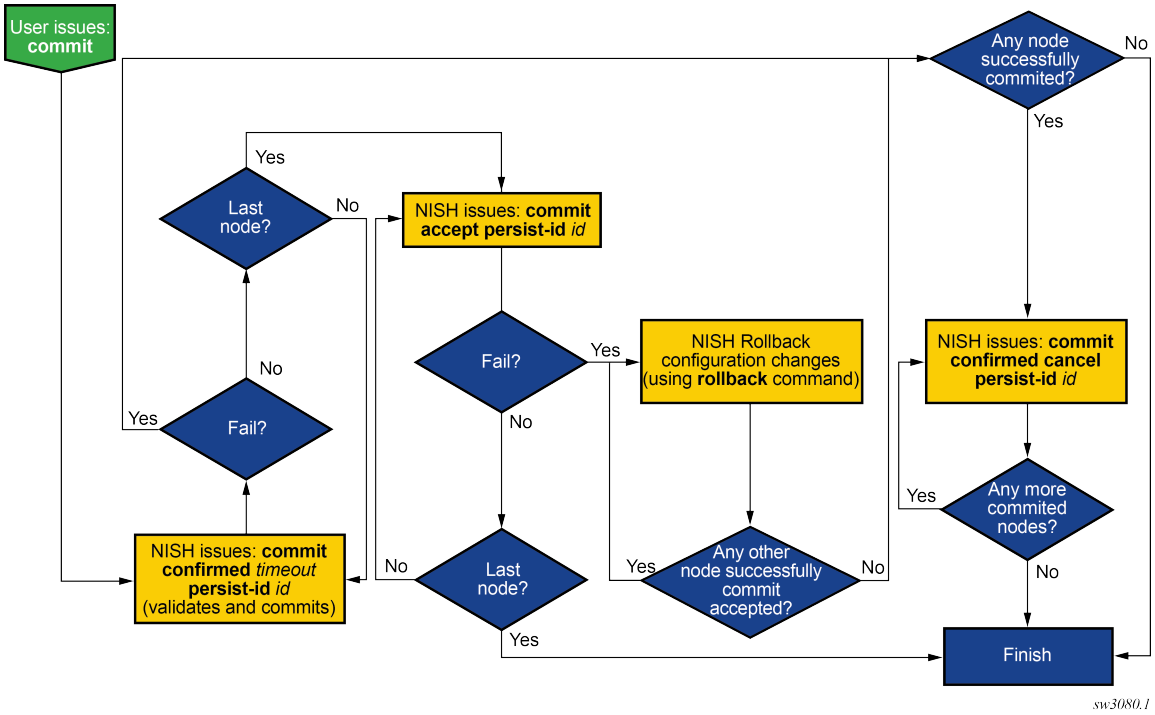
Committing configuration changes in a multinode environment from the NISH client is an atomic operation. This means that at the end of a commit operation for a specified set of nodes, every node successfully commits the configuration or all the nodes return to their configuration state before the commit.

To commit a configuration change in a multinode group, every node in the group must have automatic saving of configuration changes enabled in the MD-CLI. This is enabled by configuring the **auto-config-save** command in the **configure system management-interface cli md-cli** context on each device and committing this configuration change. The multinode commit process driven from the NISH client uses standard MD-CLI features to achieve the atomic commit. The two phases in the multinode commit operation are as follows:

1. conditional commit and validation phase (**commit confirmed** command)
2. confirmation phase (**commit confirmed accept** command)

At each stage, the nodes are processed in turn to ensure they are all in the expected state before the next stage.

Figure 4: Multinode commit process



3.11 Device name conflict resolution

The NISH client can manage multiple different devices that have the same device name. This may occur in the following situations:

- Multiple devices have the same configured system-name within the SR OS configuration.
- Multiple devices have the same device-name configured within the remote-management context.
- An SR OS device is reached through different proxies on different TCP ports.

If a device-name conflict exists when the NISH client is managing multiple devices, the device is presented to the operator in the MD-CLI in the following format.

`device-name-ip-address-tcp-port`

For example, two devices both named "mydevice" with different IP addresses appear as follows.

```
mydevice-172.16.100.3-57400
mydevice-172.16.200.5-57400
```

Note: Any device with the device name or system name "all" is always presented in the format *device-name-ip-address-tcp-port*.

3.12 Troubleshooting NISH client issues

When providing troubleshooting assistance, the Nokia support organization may request further information to diagnose issues that occur with the NISH client. This information is contained in a technical support (`tech-support`) file that the operator can generate and send to the Nokia technical support team.

To generate a `tech-support` file for NISH client issues, use the **nish admin tech-support** *filename* command.

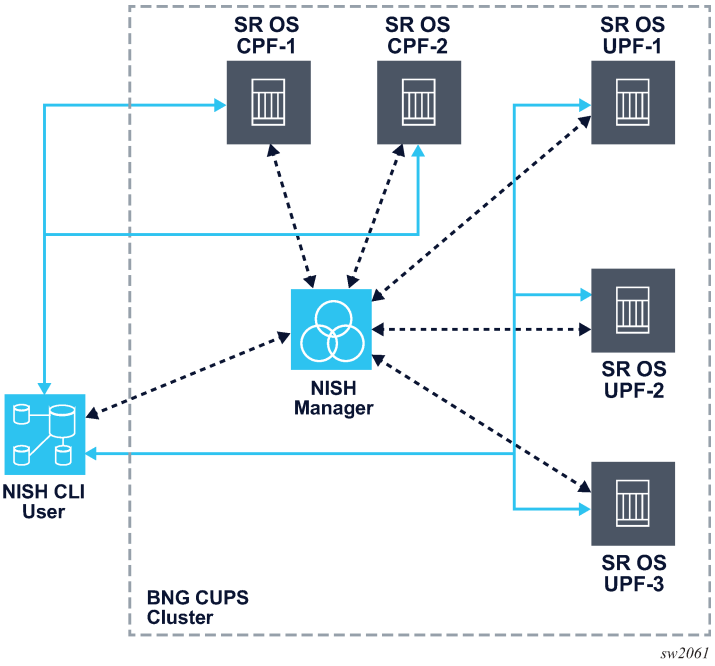
The mandatory *filename* attribute describes the relative or absolute path to a file on the local file system where the NISH client is running. The resulting encrypted file contains troubleshooting information about the NISH client. It does not provide information about the routers to which the operator is connected.

4 NISH manager

4.1 NISH manager overview

The NISH manager is an optional Linux service that provides the NISH client with a dynamic inventory of the SR OS nodes that can be managed. It correlates information about the nodes that are available. The following figure shows the inventory communication flow for the NISH client when used in conjunction with a NISH manager in a BNG CUPS deployment with two CPF nodes and three UPF nodes.

Figure 5: NISH manager overview

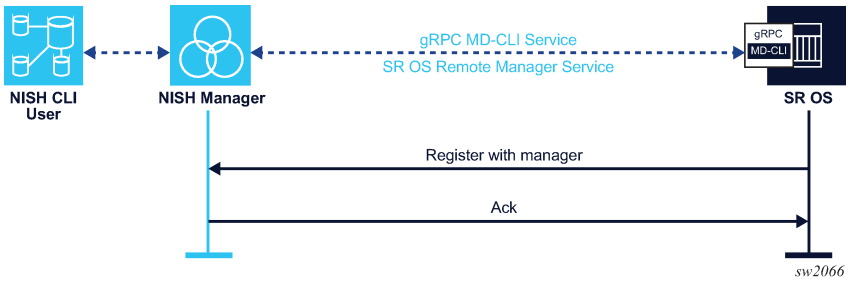


Communication with the NISH manager

SR OS nodes register with the NISH manager when configured to do so through the SR OS remote management feature. The following figure shows the SR OS remote management communication with the NISH manager.

See the *7450 ESS, 7750 SR, 7950 XRS, and VSR System Management Guide*, section "Remote Management Using a Remote Network Interface Shell Manager" for more information about the SR OS remote management feature.

Figure 6: SR OS remote management communication with the NISH manager



When a NISH client is started in manager mode, it registers with the NISH manager. The NISH manager provides the details of the available SR OS nodes to the NISH client, including names, device labels, and IP address and port information.

The following figure shows the NISH client communication with the NISH manager.

Figure 7: NISH client communication with NISH manager

The NISH manager runs as a standalone application in the foreground or background of the Linux shell or as a systemd service on the CentOS 7 Linux distribution. Using user-specified IP address and port combinations, the NISH manager listens for connections from the NISH client and the SR OS nodes that are configured with the SR OS remote management feature. By default, the NISH manager listens on all available IP interfaces and on the default gRPC port (57400).

The NISH manager does not make any outbound connections to the SR OS nodes or to the NISH client, although it does respond to both.

SR OS node state transitions in the NISH manager

When an SR OS node registers with the NISH manager, its state in the database of the NISH manager is **up**.

If the NISH manager misses the specified number of consecutive hello messages from the SR OS node, the state of the node transitions to **stale** in the database. Stale nodes appear in the NISH client but are not usable.

When the specified retention interval has passed, an SR OS node in the **stale** state transitions to the **down** state in the NISH manager database. The NISH manager removes SR OS nodes in the **down** state from its database immediately. Nodes in the **down** state are not shown.

If the number of hello messages is not configured, is set to 0, or the SR OS node does not support the state transition functionality, nodes transition from the **up** to the **down** (deleted) state when the specified retention interval has passed. They do not transition through the **stale** state.

To set the number of hello messages and the retention interval, see Command options and NISH manager configuration files.

4.2 Installing the NISH manager

About this task

The NISH manager is delivered as a set of RPM packages and is supported on the CentOS 7 Linux distribution.

The installation process installs the following:

- NISH manager in `/usr/bin/nish-manager`
- **nish-manager** as a systemd service
- End User License Agreement (EULA) in `/usr/share/licenses/nish-manager/EULA`
- Linux man pages for **nish-manager**
- log rotation schedule in `/etc/logrotate.d/nish-manager`
- CentOS **firewalld** definition in `/usr/lib/firewalld/services/nish-manager.xml`
- example configuration file in /`etc/nish-manager/nish-manager.conf`

To install the NISH manager, perform the following steps:

Procedure

Step 1. Put the RPMs in a local directory on the target machine.

Step 2. Use the YUM package manager built into CentOS 7, as shown in following example.

```
[root@server ~]# yum localinstall -y nish-manager-23.3-R1.el7.x86_64.rpm
Last metadata expiration check: 1:54:38 ago on Sun 1 Jan 2023 16:19:23 CEST.
Dependencies resolved.
================================================================================
Package Architecture Version Repository Size
================================================================================
Installing:
 nish-manager x86_64 23.3-R1.el7 @commandline 86 k
Transaction Summary
================================================================================
Install 2 Packages
Total size: 4.0 M
Installed size: 14 M
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing :
1/1 Running scriptlet: nish-manager-23.3-R1.el7.x86_64
1/1 Installing : nish-manager-23.3-R1.el7.x86_64
1/1 Running scriptlet: nish-manager-23.3-R1.el7.x86_64
1/1 Verifying : nish-manager-23.3-R1.el7.x86_64
```

```
1/1 Installed : nish-manager-23.3-R1.el7.x86_64
Complete!
```

4.3 Execution options

By default, the NISH manager is installed as a systemd Linux service and must be started to operate. The operator can also start the NISH manager as an interactive process.

4.3.1 Interactive process

When the preferred method of execution is an interactive process, the operator can start the NISH manager from the Linux command line, as shown in the following example.

```
[root@server ~]# nish-manager
2020-06-16 18:57:16.329 WARNING: Failed to read backup file: →
'/var/lib/nish-manager/connections.bkp'
2020-06-16 18:57:16.329 WARNING: Could not restore Db from: →
'/var/lib/nish-manager/connections.bkp'
2020-06-16 18:57:16.330 INFO: Server listening on [::]:57400
```

Note:

- To use the **nish-manager** command to start the NISH manager service, the user must have the required Linux permissions to bind the service to an IP address and TCP port.
- The IP address and port number specified cannot conflict with an instance of the NISH manager service that is already running. This also applies if the running instance of the NISH manager service is running within systemd.
- To run the NISH manager service in the background, append an ampersand (&) to the command. For information about additional command options, see Command options.
- The displayed warnings about the `connections.bkp` file are the expected behavior for the first run because there is no backup file yet; see Backup file.

The **nish-manager** command can also be started as an interactive process using a configuration file that contains the required variables to start an instance of the NISH manager. For more information about configuration files, see NISH manager configuration files.

To start the NISH manager as an interactive process using a configuration file, append the filename to the **nish-manager** command as shown in the following example:

```
[root@server ~]# nish-manager myconfigfile.conf
```

4.3.2 Linux service

The NISH manager is installed as a Linux systemd service, which for security reasons is installed as follows:

- in a stopped state (not running)
- disabled (does not start automatically on reboot)

The operator can start and enable the NISH manager within the systemd service at the same time or independently.

To start the NISH manager service within the systemd service and ensure that it restarts when the service or workstation is rebooted, issue the following commands.

```
[root@server ~]# systemctl enable nish-manager --now
```

The **enable** operation ensures that the NISH manager is started on system boot. The **--now** option ensures that the NISH manager service starts immediately.

It is also possible to perform the start and enable operations independently. To do so, issue the following commands.

```
[root@server ~]# systemctl start nish-manager
[root@server ~]# systemctl enable nish-manager
```

To disable and stop the NISH manager within the systemd service, issue the following command.

```
[root@server ~]# systemctl disable nish-manager --now
```

The **disable** operation ensures that NISH manager is no longer started upon reboot. The **--now** option ensures that the NISH manager service stops immediately.

To perform the stop and disable operations independently, issue the following commands.

```
[root@server ~]# systemctl stop nish-manager
[root@server ~]# systemctl disable nish-manager
```

The following example shows the `nish-manager.log` output file after startup. After the trademark, legal, and disclaimer information, NISH displays the following.

```
2020-06-16 18:52:47.952 WARNING: Failed to read backup file: →
'/var/lib/nish-manager/connections.bkp'
2020-06-16 18:52:47.952 WARNING: Could not restore Db from: →
'/var/lib/nish-manager/connections.bkp'
2020-06-16 18:52:47.953 INFO: Server listening on [::]:57400
```

Note: The displayed warnings about the `connections.bkp` file are the expected behavior for the first run because there is no backup file yet; see Backup file.

Multiple NISH manager systemd services can be started simultaneously and each one can maintain its own configuration in a NISH manager configuration file; see NISH manager configuration files for more information.

By default, when the NISH manager is started as a systemd service (for example, using **systemctl start nish-manager**) it reads from the `/etc/nish-manager/nish-manager.conf`configuration file, if this file exists. If this file does not exist, the default values are used.

The NISH manager can also be started by systemd using the values specified in a specific NISH manager configuration file. This configuration file must be located in the `/etc/nish-manager` directory and `.conf` must be appended to the file extension.

To start a systemd instance of the NISH manager using a configuration file, issue the commands as shown in the following example.

```
[root@server ~]# systemctl start nish-manager@example1
```

```
[root@server ~]# systemctl enable nish-manager@example1
```

The **start** command runs the service and reads from the configuration filename provided in the command after the @ symbol. In this example, the **nish-manager** Linux service starts and reads the configuration file `/etc/nish-manager/example1.conf` for its instantiation variables.

These commands can be combined together as shown in the following example.

```
[root@server ~]# systemctl enable nish-manager@example1 --now
```

Another systemd NISH manager service can be started by adding the configuration filename (without the `.conf` file extension) after the @ symbol. In the following example, a NISH manager service instance starts and reads the `/etc/nish-manager/example2.conf` configuration file for its instantiation variables.

```
[root@server ~]# systemctl enable nish-manager@example2 --now
```

Multiple systemd NISH manager services can run in parallel, if the IP address and TCP port combination bindings do not overlap.

4.4 Command options

The Linux man pages provide an overview of the options and flags for the **nish-manager** command.

The following table describes some of the options and flags that can be used when starting the NISH manager.

Table 4: Options and flags for the nish-manager command

Option or Flag	Description	Reference
--retention **-r**	Defines a retention or cleanup interval	See Backup file
--backup **-b**	Defines a location for the backup file	See Backup file
--ca-cert -t	Specifies a CA certificate	See Configuring NISH security
--server-cert **-k**	Provides the NISH manager server certificate	See Configuring NISH security
--server-key **-K**	Provides the NISH manager server key file	See Configuring NISH security
--log-level **-l**	Defines the log level for the NISH manager log file	See NISH manager logging

Option or Flag	Description	Reference
--listen-address **-a**	Specifies an IP address for the NISH manager to listen on [2,3,4]	—
--listen-port **-p**	Specifies a port for the NISH manager to listen on [2,3,4]	—
--stale-after **-s**	Defines the number of missed hello messages to transition a node from the up to the stale state	SR OS node state transitions in the NISH manager

4.5 Backup file

The NISH manager retains state information in a backup file. It writes the state information to the backup file for every change in the node connections.

To set the retention or cleanup time for connection registrations, use the **-r** flag or **--retention** option followed by a value in minutes. The NISH manager discards connections that have a timestamp older than the retention period (default is 60 minutes).

To set the location of the backup file, use the **-b** flag or **--backup** option followed by the absolute or relative path to the backup file. When the flag is present without a path, no backup file is created. When the flag is absent, the backup file is in the default location: `/var/lib/nish-manager/connections.bkp`.

When the NISH manager starts, it checks whether it was previously run. If a backup file is present, the NISH manager loads the previous state into its dynamic inventory.

4.6 NISH manager logging

When running as a systemd service, the NISH manager writes logs continually into a plain text log file located at `/var/log/nish-manager/nish-manager.log` on the local file system and to the local syslog service.

When running as a standalone program, the NISH manager writes logs continually to `STDOUT`.

4.6.1 Setting the verbosity level for the log file

About this task

You can set the verbosity level of the messages in the log file to the following supported options:

- **e** - error

2 Several listen addresses can be defined.

3 If the operator omits the address and port for the NISH manager to listen on, the NISH manager listens by default on all IP interfaces available and on the default gRPC port (57400).

4 Multiple instances of the NISH manager can run on a single server or workstation, if the IP address and port combinations of the instances do not clash.

- **w** - warning
- **i** - info
- **d** - debug

Procedure

Enter the **nish-manager** command with the **-l** flag or **--log-level** option and the verbosity level setting.

Example

nish-manager [-l | --log-level] *level*

```
nish-manager -l w
```

4.6.2 Changing the syslog facility

About this task

The facility that the NISH manager sends to the syslog service can be configured in the NISH manager configuration file.

Perform the following steps to change the syslog facility:

Procedure

Step 1. Change the value of the NISH_MANAGER_SYSLOG_FACILITY option in the NISH manager configuration file. See NISH manager configuration files.

Step 2. Restart the NISH manager.

4.7 NISH manager configuration files

The NISH manager provides the ability to place commonly used attributes into a configuration file that is provided when the NISH manager starts. The configuration file can be used when executing the NISH manager as an interactive process or as a Linux service.

Note: When a configuration file is provided, the user does not need to manually provide any variables on the command line. However, if variables are provided on the command line, they overwrite the variables provided in the configuration file.

After the NISH manager installation is completed, an example (and default) configuration file can be found in `/etc/nish-manager/nish-manager.conf`.

When executing the **nish-manager** command as a systemd service or as an interactive service, the configuration file must be placed in the `/etc/nish-manager` directory. This is the only location that the system checks for the file.

The following table describes the options that are supported in a NISH manager configuration file.

Table 5: Supported variables in NISH manager configuration file

Variable	Description	Multiple options per file
NISH_MANAGER_RETENTION	Time that an SR OS node or NISH client connection is retained as a valid node within the NISH manager without an update message, before it is expunged from the NISH manager	No
NISH_MANAGER_LISTEN_ ADDRESS	IPv4 or IPv6 address to which the NISH manager binds; if multiple addresses are required, multiple NISH_ MANAGER_LISTEN_ADDRESS lines must be provided, with one address per line Default is "::", which indicates to bind to all IPv4 and IPv6 addresses present on the system	Yes
NISH_MANAGER_LISTEN_ PORT	TCP port to which the NISH manager binds; if multiple ports are required, multiple NISH_ MANAGER_ LISTEN_PORT lines must be provided with one port per line Default 57400 is used unless a NISH_ MANAGER_ LISTEN_PORT line is supplied	Yes
NISH_MANAGER_BACKUP_ FILE	Path to the filename that the NISH manager service reads from and writes to, for a persistent cache of connected SR OS nodes and NISH clients Default is `/var/lib/nish-manager/connections.bkp`	No
NISH_MANAGER_LOG_LEVEL	Logging verbosity level of the NISH manager process Default is "i" (informational)	No
NISH_MANAGER_SYSLOG_ FACILITY	Syslog facility setting for logging data Default is "local7"	No
NISH_MANAGER_CA_CERT	Path to the CA TLS certificate	No
NISH_MANAGER_SERVER_ CERT	Path to the server TLS certificate	No
NISH_MANAGER_SERVER_ KEY	Path to the service TLS key	No
NISH_MANAGER_STALE_ HELLO_COUNT	The number of missed hello messages to transition a node from the up to the stale state	No

It is permitted to provision multiple variables in the configuration file for the NISH manager listen address and listen port. The following table lists example address/port bindings when multiple variables are used in the configuration file.

Table 6: Example address to port bindings for configuration file variable

Example configuration file variables	Example address to port bindings
NISH_MANAGER_LISTEN_ADDRESS=1.1.1.1	1.1.1.1:57401
NISH_MANAGER_LISTEN_PORT=57401	1.1.1.1:57402
NISH_MANAGER_LISTEN_ADDRESS=2.2.2.2	2.2.2.2:57402
NISH_MANAGER_LISTEN_PORT=57402	2.2.2.2:57403

5 Configuring NISH security

Prerequisites

This chapter provides an example of a NISH security configuration. The NISH client and the NISH manager use gRPC, which operates over the HTTP2 transport and can make use of TLS encryption. The default gRPC TCP port is 57400.

To use TLS, the following certificates and keys are required:

- Certificate Authority (CA) certificate
- Node certificate that contains all DNS names and IP addresses present on the nodes that are used to manage the device
- Node certificate key
- NISH manager certificate that contains all DNS names and IP addresses to which all NISH manager processes are bound
- NISH manager certificate key

Note: The use of TLS is strongly recommended for all operational deployments. In lab environments, the NISH client and NISH managers can operate without TLS.
All certificates must be signed by the provided Certificate Authority.

To start the NISH client in TLS mode, add the **-t** flag or **--ca-cert** option followed by the relative or absolute path to the CA certificate file; for example, `cacert.pem`.

Note: A single CA certificate is supported.

To start the NISH manager in TLS mode, add the following flags or options:

- **-k** or **--server-cert**, followed by the relative or absolute path to the NISH manager certificate file; for example, `nish_manager_cert.pem`
- **-K** or **--server-key**, followed by the relative or absolute path to the NISH manager certificate key file; for example, `nish_manager_key.pem`
- **-t** or **--ca-cert**, followed by the relative or absolute path to the CA certificate file; for example, `cacert.pem`

The certificates must be copied in the SR OS nodes to successfully establish a secured connection.

Note: See the *7450 ESS, 7750 SR, 7950 XRS, and VSR System Management Guide* for more information about TLS and certificates.

The following procedure is an example of how to install the certificates for use with the NISH client and the NISH manager. The example commands in the procedure assume the following filenames for the certificates:

- `node_cert.pem` for the node certificate
- `node_key.pem` for the node certificate key

- `cacert.pem` for the CA certificate

Note: When installing certificates, operators must use commands appropriate for their encryption environment.

The following are example steps for NISH security configuration:

Procedure

Step 1. Copy the SR OS node certificate, the node certificate key, and the CA certificate to the SR OS node.

Step 2. Import the certificates into the SR OS certificate manager.

```
//admin certificate import type cert input cf3:/node_cert.pem
# output node.cert format pem
//admin certificate import type key input cf3:/node_key.pem
# output node.key format pem
//admin certificate import type cert input cf3:/cacert.pem
# output cacert.pem format pem
```

Note: The `//` denotes that these SR OS commands are executed in the classic CLI environment.

Step 3. Configure the certificate profile.

```
/configure system security tls cert-profile "grpc_cert_profile" admin-state enable
/configure system security tls cert-profile "grpc_cert_profile" entry 1
# certificate-file "node.cert"
/configure system security tls cert-profile "grpc_cert_profile" entry 1
# key-file "node.key"
```

Step 4. Configure the TLS cipher list.

```
/configure system security tls server-cipher-list "all" { }
/configure system security tls server-cipher-list "all" cipher 1 name
# tls-rsa-with3des-ede-cbc-sha
/configure system security tls server-cipher-list "all" cipher 2 name
# tls-rsa-with-aes128-cbc-sha
/configure system security tls server-cipher-list "all" cipher 3 name
# tls-rsa-with-aes128-cbc-sha256
/configure system security tls server-cipher-list "all" cipher 4 name
# tls-rsa-with-aes256-cbc-sha
/configure system security tls server-cipher-list "all" cipher 5 name
# tls-rsa-with-aes256-cbc-sha256
```

Step 5. Configure the TLS server profile.

```
/configure system security tls server-tls-profile "grpc_tls_profile"
# admin-state enable
/configure system security tls server-tls-profile "grpc_tls_profile"
# cert-profile "grpc_cert_profile"
/configure system security tls server-tls-profile "grpc_tls_profile"
# cipher-list "all"
```

Step 6. Configure the client certificate.

```
/configure system security pki ca-profile "ca" admin-state enable
/configure system security pki ca-profile "ca" cert-file "cacert.pem"
```

```
/configure system security pki ca-profile "ca" revocation-check crl-optional
/configure system security tls trust-anchor-profile "grpc_ca" { }
/configure system security tls trust-anchor-profile "grpc_ca" { trust-anchor "ca"}
```

Step 7. Configure the client cipher list.

```
/configure system security tls client-cipher-list "all" { }
/configure system security tls client-cipher-list "all" cipher 1 name
# tls-rsa-with3des-ede-cbc-sha
/configure system security tls client-cipher-list "all" cipher 2 name
# tls-rsa-with-aes128-cbc-sha
/configure system security tls client-cipher-list "all" cipher 3 name
# tls-rsa-with-aes128-cbc-sha256
/configure system security tls client-cipher-list "all" cipher 4 name
# tls-rsa-with-aes256-cbc-sha
/configure system security tls client-cipher-list "all" cipher 5 name
# tls-rsa-with-aes256-cbc-sha256
```

Step 8. Configure the client TLS profile.

```
/configure system security tls client-tls-profile "grpc_tls_client_profile" →
admin-state enable
/configure system security tls client-tls-profile "grpc_tls_client_profile" →
cipher-list "all"
/configure system security tls client-tls-profile "grpc_tls_client_profile" →
trust-anchor-profile "grpc_ca"
```

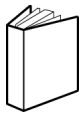
Step 9. Enable TLS protection for gRPC.

```
/configure system grpc tls-server-profile "grpc_tls_profile"
```

Step 10. Enable TLS protection for the remote management service (or per manager).

```
/configure system management-interface remote-management client-tls-profile
# "grpc_tls_client_profile"
```

Customer document and product support



Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback