ipinfusion™

OcNOS®

Open Compute Network Operating System for Service Providers

NetConf User Manual

Version 7.0.1

July 2026

IP Infusion Inc. Proprietary

©2026 IP Infusion Inc. All Rights Reserved.

This documentation is subject to change without notice. The software described in this document and this documentation are furnished under a license agreement or nondisclosure agreement. The software and documentation may be used or copied only in accordance with the terms of the applicable agreement. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's internal use without the written permission of IP Infusion Inc.

IP Infusion Inc.

3979 Freedom Circle, Suite 900

Santa Clara, CA 95054

+1 408-400-1900

http://www.ipinfusion.com/

For support, questions, or comments via E-mail, contact:

support@ipinfusion.com

Trademarks:

IP Infusion and are trademarks or registered trademarks of IP Infusion. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Use of certain software included in this equipment is subject to the IP Infusion, Inc. End User License Agreement at http://www.ipinfusion.com/license. By using the equipment, you accept the terms of the End User License Agreement.

CONTENTS

Contents ... **3**

Preface ... **9**
About this Guide ... 9
Audience ... 9
Conventions ... 9
IP Infusion Product Release Version ... 9
Related Documentation ... 10
Feature Availability ... 10
Migration Guide ... 10
IP Maestro Support ... 10
Technical Support ... 10
Technical Sales ... 10
Technical Documentation ... 10
Documentation Disclaimer ... 11
Comments ... 11
Command Line Interface ... 12
Overview ... 12
Chapter Organization ... 12
Command Line Interface Help ... 12
Command Completion ... 13
Command Abbreviations ... 13
Command Line Errors ... 14
Command Negation ... 14
Syntax Conventions ... 14
Variable Placeholders ... 15
Command Description Format ... 16
Keyboard Operations ... 16
Show Command Modifiers ... 17
Begin Modifier ... 17
Include Modifier ... 18
Exclude Modifier ... 18
Redirect Modifier ... 19
Last Modifier ... 19
String Parameters ... 19
Command Modes ... 20
Command Mode Tree ... 21
Transaction-based Command-line Interface ... 21

Introduction ... **23**

NetConf Quick Start ... **24**
NetConf Clients ... 24

© 2026 IP Infusion Inc. Proprietary

Periodic GET Caching 24
Install Yangcli 25
Download Yang Files 25

Yangcli Operations 26

Establish Connection 26
Configure the Device 27
Limitations 28
edit-config Operation 28
Default-Operations 29
Configuration 30
Retrieve Candidate Configuration 30
Limitations 30
Commit Candidate Configuration 32
Retrieve Running Configuration 32
Retrieve Running Configuration and State Data 32
Copy Running Configuration to Startup 33
YANG Unique Datamodel Support 33
Limitations 34
Error Messages 34
Warning Messages 35
Candidate Configuration Store Lock 36
Startup configuration Store Lock 37
Running Configuration Store Lock 37
Force Unlock 38
Sys-reload 39
Example 39
Sys-shutdown 39
Example 39
Custom RPC Support in OcNOS 39
CLI-based Configuration RPC 39
Load configuration RPC 39
Copy-text configuration 42
Custom show RPC 44
custom-cmlsh-show-rpc 44
Candidate Difference RPC 48
netconf-config-diff RPC 48
yang RPC 48
Backup-configuration 50
create-backup-configuration 50
show-backup-configurations 50
restore-backup-configuration 51
Feature-netconf 51
set-feature-netconf-status 51
get-feature-netconf-status 52
Transaction-limit 52
set-transaction-limit 52

© 2026 IP Infusion Inc. Proprietary

Example 53
show-transaction-limit 53
Example 53
Auto-config-sync 53
Example 54
Backend API-support for Custom GET/SET RPC 54
GET (Read) RPC 54
SET (Write) RPC 56

Transactions 58
Discard Changes 58
Commit 59
Confirmed-commit 60
commit confirmed 60
cancel-commit 60
commit confirmed persist= <id> 61
commit 62
Transaction Limit 63
Set Transaction Limit 63
View Transaction Limit 63

NetConf Monitoring 64
Retrieve YANG Schema 64
Error Messages 64
Retrieving Schema List 65

URL Capabilities 66
Overview 66
Limitations 66
Prerequisites 66
Copy-config 66
Supported Source / Target Combinations: 67
From HTTP to the Candidate Configuration Store 67
From HTTPs to the Candidate Configuration Store 67
From FTP to the Candidate Configuration Store 68
From SFTP to the Candidate Configuration Store 68
From a Local file to the Candidate Configuration Store 69
From the Candidate Configuration Store to HTTP 70
From the Candidate Configuration Store to HTTPS 70
From the Candidate Configuration Store to FTP 70
From the Candidate Configuration Store to SFTP 71
From the Candidate Configuration Store to the Local File 71
Copy-config Error Messages 72
Edit-config 72
Edit-config using HTTP 73
Edit-config using FTP 73
Edit-config using local file 74
Edit-config Error Messages 75

© 2026 IP Infusion Inc. Proprietary

Delete-config 75
Delete configuration from HTTP server 75
Delete configuration from FTP 75
Delete local file 76
Delete-config Error Messages 76
copy-text-config 77
Feature Characteristics 77
Supported Source and Target 77
RPC Format 78
Parameters 78
Example 78
Local File Path 78

Event Notification 79
Notifications for Operational Status 79
NETCONF Notification Subscription 79
Basic Subscription 79
Enhanced Subscription Filtering (Severity, Event Class, Stream) 80
Event Classes and Applicable Notifications 80
Severity & Event Class Filter 80
Stream Filter 80
Summary 81
netconf-config-change 81
Example 82
netconf-capability-change 83
netconf-session-start 83
netconf-session-end 83
Notification Cache Support in CML 84
Behavior of the feature when enabled: 84
Priority-based Caching: 84

Validate Capability 84

With-defaults Capability 86
Explicit Mode 86
Trim Mode 86
Report-All Mode 86
Report-All-Tagged Mode 86
Edit-config Behavior 87

Supported Operations 88

Sys-Update using NetConf 92
Download the Image and Install 92
On the Switch 92
Validation 92
To Delete the Downloaded Image 93
On the Switch 93

© 2026 IP Infusion Inc. Proprietary

Validation 93
To Cancel the Image Download 93
On the Switch 93
Validation 94

SSH Client 95
Establish a Connection 95
Send Client Help Message to NetConf Server 95
NETCONF-SSH over User Defined VRF 96
Server Configuration for User Defined VRFs 96
Server Configuration for User Defined VRFs to Configure SSH Ports 97
Validation 97

OpenDaylight Controller 98
Installation 98
Connecting with an OcNOS Device 98
Restconf Endpoints 99
Patch or Merge Operation 100
POSTMAN 102
Installation 102
Edit-config 103
Edit-config using API-DOCS 103
Edit-config using Terminal 103
Edit-config using POSTMAN 104
Known Issues and Troubleshooting 105
Mounted resources on API-Doc showing 500:internal server error 105
To apply the PATCH using .JAR files 105
Unavailable-capabilities in Hello Message 105
JAVA_HOME Not Set 105
OcNOS Device Compatibility 106

NetConf Over Transport Layer Security 107
Topology 107
Generating TLS Certificates 107
Generate CA Certificates 108
Generate Server Certificates 108
Generate Client Certificates 109
Enabling NETCONF Server over SSH and TLS 110
Installing Netopeer2 110
Creating a new Docker Image (Netopeer2) 111
Using Netopeer2 as NETCONF Client (Over TLS) 112
NETCONF-TLS over User Defined VRFs 113
Server Configuration for User Defined VRFs 113
Server Configuration for User Defined VRFs to Configure TLS Ports 113
Validation 114

Call Home 115
Configuration 115

© 2026 IP Infusion Inc. Proprietary

NetConf Access Control Model User Guide 116
Overview 116
Feature Characteristics 116
Configuration 117
Initial NACM Configuration 117
NACM Rules types 118
Benefits 118
Prerequisites 118
Common NACM Rule Fields 118
Valid Path Notes for NETCONF NACM <path> Rules 120
Creating NetConf RPC for NACM 121
RPC Configurations for NACM 121
Purpose of Rules 127
Rules Breakdown 127
Rule Order 128
Best Practice 128
Rule Insertion Control 128
Enforce Access Control for OpenConfig Data Models 129
Ordered Rule Management with Yang:Insert 129
Using NETCONF to Manage NACM Rules 130
Troubleshooting 130
Limitations 131
Glossary 131

Image Upgrade by Traffic Diversion (IUTD) 132
Overview 132
Feature Characteristics 132
Prerequisites 132
Sequence of IUTD process 133
Initial Scenario & Traffic Diversion (Service UP -> DOWN) 133
Installation and OS Update 133
Verification and Traffic Restoration 133
Judgement Criteria 135
Topology 135
In-Band Management Scenario 135
Judgement logic for detecting changes in service status - Leaf 135
Judgement logic for detecting changes in service status - Spine 136
Out-Band Management Scenario 139
Datamodel for Service Tracking 140
Validation 142

© 2026 IP Infusion Inc. Proprietary

PREFACE

About this Guide

This guide describes how to configure NetConf User Manual in OcNOS.

Audience

This guide is intended for network administrators and other engineering professionals who configure OcNOS.

Conventions

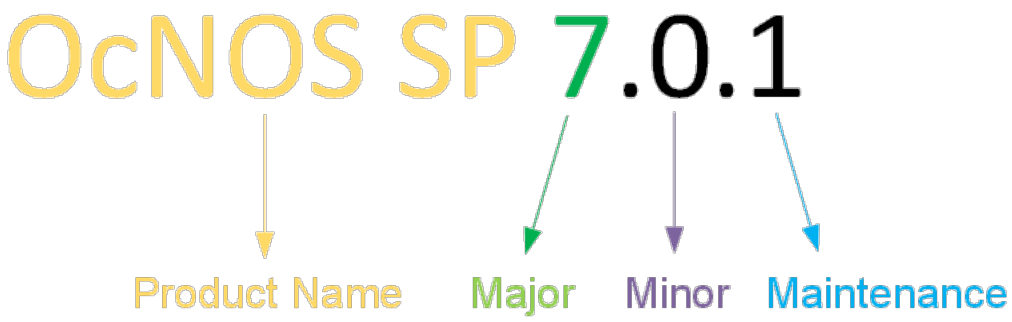
The Table 1 table shows the conventions used in this guide.

Table 1. Conventions

Convention	Description
Italics	Emphasized terms; titles of books
Note:	Special instructions, suggestions, or warnings
`monospaced type`	Code elements such as commands, parameters, files, and directories

IP Infusion Product Release Version

Each integer in release numbers indicates Major, Minor, and Maintenance release versions. Build numbers that follow the release numbers are for internal tracking and verification of the software build process and are visible to customers as part of the software version number.



Product Name: IP Infusion Product Family

Major Version: New customer-facing functionality that represents a significant change to the code base; including a significant marketing change or direction in the product.

© 2026 IP Infusion Inc. Proprietary

Minor Version: Enhancements or extensions to existing features, changes to address external needs, or internal improvements to satisfy new sales regions or marketing initiatives.

Maintenance Version: A collection of product bugs or issues usually scheduled every 30 or 60 days, based on the number of issues.

Related Documentation

For information about installing OcNOS, see the *Installation Guide* for your platform.

Feature Availability

Each OcNOS SKU contains a set of supported features. For a list of available features based on the SKU that you purchased, refer to the *Feature Matrix* .

Migration Guide

Check the *Migration Guide* for necessary configuration changes before migrating from one version of OcNOS to another.

IP Maestro Support

Monitor devices running OcNOS Release 6.3.4-70 and above using IP Maestro software.

Technical Support

IP Infusion maintains an online technical support site that provides a variety of technical support programs for licensed OcNOS customers at the Technical Assistance Center.

Customers and partners enjoy full access to the support website. The site allows customers and partners to open technical support calls, update open calls with new information, and review the status of open or closed calls. The password-protected site includes technical documentation, Release Notes, and descriptions of service offerings.

Technical Sales

Contact the IP Infusion sales representative for more information about the OcNOS solution.

Technical Documentation

For core commands and configuration procedures, visit: Product Documentation.

For training videos, visit: OcNOS Free Training Videos.

For a list of supported platforms and SKUs of OcNOS features, refer to the OcNOS Feature Matrix.

© 2026 IP Infusion Inc. Proprietary

Documentation Disclaimer

The global documentation site is evolving to provide an enhanced website user experience for select topics included in this release. Some guides are now available outside the existing documentation library and can be accessed directly from custom documentation landing pages. These guides offer robust in-built search functionality.

For the latest documentation, visit the product-specific documentation landing page and select the relevant guide.

Comments

If you have comments, or need to report a problem with the content, contact techpubs@ipinfusion.com.

© 2026 IP Infusion Inc. Proprietary

Command Line Interface

This chapter introduces the OcNOS Command Line Interface (CLI) and how to use its features.

Overview

You use the CLI to configure, monitor, and maintain OcNOS devices. The CLI is text-based and each command is usually associated with a specific task.

You can give the commands described in this manual locally from the console of a device running OcNOS or remotely from a terminal emulator such as putty or xterm. You can also use the commands in scripts to automate configuration tasks.

Chapter Organization

The chapters in command references are organized as described in Command Description Format (page 16).

The chapters in configuration guides are organized into these major sections:

- An overview that explains a configuration in words
- Topology with a diagram that shows the devices and connections used in the configuration
- Configuration steps in a table for each device where the left-hand side shows the commands you enter and the right-hand side explains the actions that the commands perform
- Validation which shows commands and their output that verify the configuration

Command Line Interface Help

You access the CLI help by entering a full or partial command string and a question mark “?”. The CLI displays the command keywords or parameters along with a short description. For example, at the CLI command prompt, type:

```
> show ?
```

The CLI displays this keyword list with short descriptions for each keyword:

```
show ?
  application-priority            Application Priority
  arp                             Internet Protocol (IP)
  bfd                             Bidirectional Forwarding Detection (BFD)
  bgp                             Border Gateway Protocol (BGP)
  bi-lsp                          Bi-directional lsp status and configuration
  bridge                          Bridge group commands
  ce-vlan                         COS Preservation for Customer Edge VLAN
  class-map                       Class map entry
  cli                             Show CLI tree of current mode
  clns                            Connectionless-Mode Network Service (CLNS)
  control-adjacency               Control Adjacency status and configuration
  control-channel                 Control Channel status and configuration
  cspf                            CSPF Information
  customer                        Display Customer spanning-tree
  cvlan                           Display CVLAN information
  debugging                       Debugging functions
  etherchannel                    LACP etherchannel
```

```
  ethernet                      Layer-2
...
```

If you type the ? in the middle of a keyword, the CLI displays help for that keyword only.

```
> show de?
debugging  Debugging functions
```

If you type the ? in the middle of a keyword, but the incomplete keyword matches several other keywords, OcNOS displays help for all matching keywords.

```
> show i? (CLI does not display the question mark).
interface  Interface status and configuration
ip         IP information
isis       ISIS information
```

Command Completion

The CLI can complete the spelling of a command or a parameter. Begin typing the command or parameter and then press the tab key. For example, at the CLI command prompt type **sh**:

```
> sh
```

Press the tab key. The CLI displays:

```
> show
```

If the spelling of a command or parameter is ambiguous, the CLI displays the choices that match the abbreviation. Type **show i** and press the tab key. The CLI displays:

```
> show i
  interface  ip         ipv6       isis
> show i
```

The CLI displays the **interface** and **ip** keywords. Type **n** to select **interface** and press the tab key. The CLI displays:

```
> show in
> show interface
```

Type ? and the CLI displays the list of parameters for the **show interface** command.

```
> show interface
  IFNAME  Interface name
  |       Output modifiers
  >       Output redirection
  <cr>
```

The CLI displays the only parameter associated with this command, the **IFNAME** parameter.

Command Abbreviations

The CLI accepts abbreviations that uniquely identify a keyword in commands. For example:

```
> sh int xe0
```

is an abbreviation for:

```
> show interface xe0
```

© 2026 IP Infusion Inc. Proprietary

Command Line Errors

Any unknown spelling causes the CLI to display the error **Unrecognized command** in response to the ?. The CLI displays the command again as last entered.

```
> show dd?
% Unrecognized command
> show dd
```

When you press the Enter key after typing an invalid command, the CLI displays:

```
(config)#router ospf here
                        ^
% Invalid input detected at '^' marker.
```

where the ^ points to the first character in error in the command.

If a command is incomplete, the CLI displays the following message:

```
> show
% Incomplete command.
```

Some commands are too long for the display line and can wrap mid-parameter or mid-keyword, as shown below. This does *not* cause an error and the command performs as expected:

```
area 10.10.0.18 virtual-link 10.10.0.19 authentication-key 57393
```

Command Negation

Many commands have a **no** form that resets a feature to its default value or disables the feature. For example:

- The **ip address** command assigns an IPv4 address to an interface
- The **no ip address** command removes an IPv4 address from an interface

Syntax Conventions

Table 2 describes the conventions used to represent command syntax in this reference.

Table 2. Syntax conventions

Convention	Description	Example
monospaced font	Command strings entered on a command line	**show ip ospf**
lowercase	Keywords that you enter exactly as shown in the command syntax.	`show ip ospf`
`UPPERCASE`	See Variable Placeholders (page 15)	`IFNAME`
()	Optional parameters, from which you must select one. Vertical bars delimit the selections. Do not enter the parentheses or vertical bars as part of the command.	`(A.B.C.D|<0-4294967295>)`
()	Optional parameters, from which you select one or none. Vertical bars delimit the	`(A.B.C.D|<0-4294967295>|)`

© 2026 IP Infusion Inc. Proprietary

Table 2. Syntax conventions (continued)

Convention	Description	Example
	selections. Do not enter the parentheses or vertical bars as part of the command.	
`()`	Optional parameter which you can specify or omit. Do not enter the parentheses or vertical bar as part of the command.	`(IFNAME|)`
`{}`	Optional parameters, from which you must select one or more. Vertical bars delimit the selections. Do not enter the braces or vertical bars as part of the command.	`{intra-area <1-255>|inter-area <1-255>|external <1-255>}`
`[]`	Optional parameters, from which you select zero or more. Vertical bars delimit the selections. Do not enter the brackets or vertical bars as part of the command.	`[<1-65535>|AA:NN|internet|local-AS|no-advertise|no-export]`
`?`	Nonrepeatable parameter. The parameter that follows a question mark can only appear once in a command string. Do not enter the question mark as part of the command.	`?route-map WORD`
`.`	Repeatable parameter. The parameter that follows a period can be repeated more than once. Do not enter the period as part of the command.	`set as-path prepend .<1-65535>`

Variable Placeholders

Table 3 shows the tokens used in command syntax use to represent variables for which you supply a value.

Table 3. Variable placeholders

Token	Description
`WORD`	A contiguous text string (excluding spaces)
`LINE`	A text string, including spaces; no other parameters can follow this parameter
`IFNAME`	Interface name whose format varies depending on the platform; examples are: **`eth0`**, **`Ethernet0`**, **`ethernet0`**, **`xe0`**
`A.B.C.D`	IPv4 address
`A.B.C.D/M`	IPv4 address and mask/prefix
`X:X::X:X`	IPv6 address
`X:X::X:X/M`	IPv6 address and mask/prefix
`HH:MM:SS`	Time format
`AA:NN`	BGP community value

© 2026 IP Infusion Inc. Proprietary

Table 3. Variable placeholders (continued)

Token	Description
XX:XX:XX:XX:XX:XX	MAC address
<1-5> <1-65535> <0-2147483647> <0-4294967295>	Numeric range

Command Description Format

The Table 4 table explains the sections used to describe each command in this reference.

Table 4. Command descriptions

Section	Description
Command Name	The name of the command, followed by what the command does and when should it be used
Command Syntax	The syntax of the command
Parameters	Parameters and options for the command
Default	The state before the command is executed
Command Mode	The mode in which the command runs; see Command Modes (page 20)
Applicability	The command introduced in a specific release version and modified or updated in subsequent versions.
Example	An example of the command being executed

Keyboard Operations

The Table 5 table lists the operations you can perform from the keyboard.

Table 5. Keyboard operations

Key combination	Operation
Left arrow or Ctrl+b	Moves one character to the left. When a command extends beyond a single line, you can press left arrow or Ctrl+b repeatedly to scroll toward the beginning of the line, or you can press Ctrl+a to go directly to the beginning of the line.
Right arrow or Ctrl-f	Moves one character to the right. When a command extends beyond a single line, you can press right arrow or Ctrl+f repeatedly to scroll toward the end of the line, or you can press Ctrl+e to go directly to the end of the line.
Esc, b	Moves back one word
Esc, f	Moves forward one word
Ctrl+e	Moves to end of the line

© 2026 IP Infusion Inc. Proprietary

Table 5. Keyboard operations (continued)

Key combination	Operation
Ctrl+a	Moves to the beginning of the line
Ctrl+u	Deletes the line
Ctrl+w	Deletes from the cursor to the previous whitespace
Alt+d	Deletes the current word
Ctrl+k	Deletes from the cursor to the end of line
Ctrl+y	Pastes text previously deleted with Ctrl+k, Alt+d, Ctrl+w, or Ctrl+u at the cursor
Ctrl+t	Transposes the current character with the previous character
Ctrl+c	Ignores the current line and redisplays the command prompt
Ctrl+z	Ends configuration mode and returns to exec mode
Ctrl+l	Clears the screen
Up Arrow or Ctrl+p	Scroll backward through command history
Down Arrow or Ctrl+n	Scroll forward through command history

Show Command Modifiers

Note: The show command output included in the guides is for illustration purposes only. Based on the combination of features enabled and ongoing enhancements made to the commands, the output for these commands may vary. For instance, the actual command output may differ depending on the software version, configuration, and platform. Field names, values, and formats are subject to change.

You can use two tokens to modify the output of a **show** command. Enter a question mark to display these tokens:

```
# show users ?
  | Output modifiers
  > Output redirection
```

You can type the | (vertical bar character) to use output modifiers. For example:

```
> show rsvp | ?
  begin     Begin with the line that matches
  exclude   Exclude lines that match
  include   Include lines that match
  last      Last few lines
  redirect  Redirect output
```

Begin Modifier

The **begin** modifier displays the output beginning with the first line that contains the input string (everything typed after the **begin** keyword). For example:

```
# show running-config | begin xe1
...skipping
interface xe1
ipv6 address fe80::204:75ff:fee6:5393/64
!
interface xe2
```

```
ipv6 address fe80::20d:56ff:fe96:725a/64
!
line con 0
login
!
end
```

You can specify a regular expression after the **begin** keyword, This example begins the output at a line with either “xe2” or “xe4”:

```
# show running-config | begin xe[2-4]

...skipping
interface xe2
 shutdown
!
interface xe4
 shutdown
!
interface svlan0.1
 no shutdown
!
route-map myroute permit 2
!
route-map mymap1 permit 10
!
route-map rmap1 permit 2
!
line con 0
 login
line vty 0 4
 login
!
end
```

Include Modifier

The **include** modifier includes only those lines of output that contain the input string. In the output below, all lines containing the word “input” are included:

```
# show interface xe1 | include input
    input packets 80434552, bytes 2147483647, dropped 0, multicast packets 0
    input errors 0, length 0, overrun 0, CRC 0, frame 0, fifo 1, missed 0
```

You can specify a regular expression after the **include** keyword. This examples includes all lines with “input” or “output”:

```
#show interface xe0 | include (in|out)put
    input packets 597058, bytes 338081476, dropped 0, multicast packets 0
    input errors 0, length 0, overrun 0, CRC 0, frame 0, fifo 0, missed 0
    output packets 613147, bytes 126055987, dropped 0
    output errors 0, aborted 0, carrier 0, fifo 0, heartbeat 0, window 0
```

Exclude Modifier

The **exclude** modifier excludes all lines of output that contain the input string. In the following output example, all lines containing the word “input” are excluded:

```
# show interface xe1 | exclude input
Interface xe1
  Scope: both
  Hardware is Ethernet, address is 0004.75e6.5393
  index 3 metric 1 mtu 1500 <UP,BROADCAST,RUNNING,MULTICAST>
```

```
  VRF Binding: Not bound
  Administrative Group(s): None
  DSTE Bandwidth Constraint Mode is MAM
  inet6 fe80::204:75ff:fee6:5393/64
    output packets 4438, bytes 394940, dropped 0
    output errors 0, aborted 0, carrier 0, fifo 0, heartbeat 0, window 0
    collisions 0
```

You can specify a regular expression after the **exclude** keyword. This example excludes lines with “output” or “input”:

```
show interface xe0 | exclude (in|out)put
Interface xe0
  Scope: both
  Hardware is Ethernet  Current HW addr: 001b.2139.6c4a
  Physical:001b.2139.6c4a  Logical:(not set)
  index 2 metric 1 mtu 1500 duplex-full arp ageing timeout 3000
  <UP,BROADCAST,RUNNING,MULTICAST>
  VRF Binding: Not bound
  Bandwidth 100m
  DHCP client is disabled.
  inet 10.1.2.173/24 broadcast 10.1.2.255
  VRRP Master of :  VRRP is not configured on this interface.
  inet6 fe80::21b:21ff:fe39:6c4a/64
    collisions 0
```

Redirect Modifier

The **redirect** modifier writes the output into a file. The output is not displayed.

```
# show cli history | redirect /var/frame.txt
```

The output redirection token (>) does the same thing:

```
# show cli history >/var/frame.txt
```

Last Modifier

The **last** modifier displays the output of last few number of lines (As per the user input). The last number ranges from 1 to 9999.

For example:

```
#show running-config | last 10
```

String Parameters

The restrictions in Table 6 apply for all string parameters used in OcNOS commands, unless some other restrictions are noted for a particular command.

Table 6. String parameter restrictions

Restriction	Description
Input length	1965 characters or less
Restricted special characters	“?”, “,”, “>”, “|”, and “=” The “|” character is allowed only for the **description** command in interface mode.

Command Modes

Commands are grouped into modes arranged in a hierarchy. Each mode has its own set of commands. The table below lists the command modes common to all protocols.

Table 7. Common Command Modes

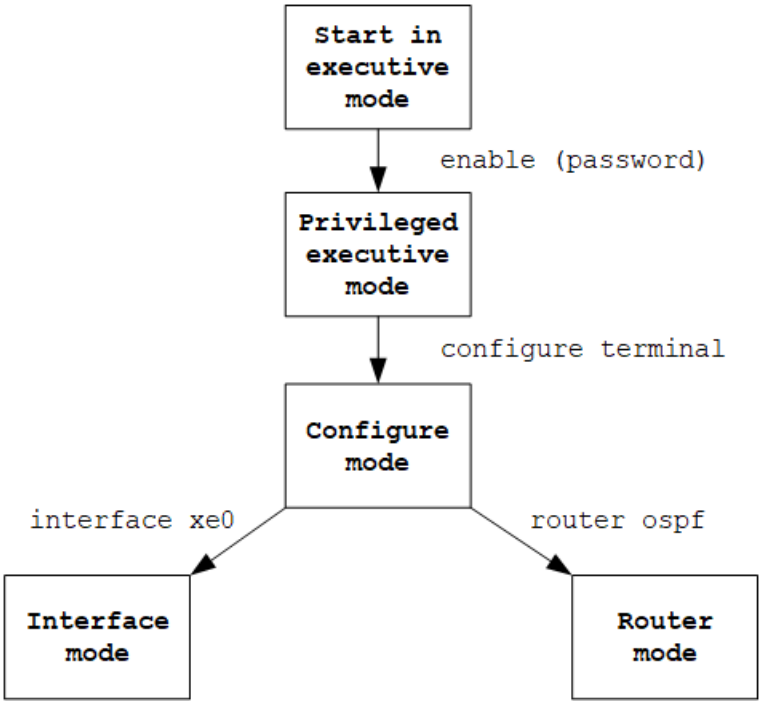
Name	Description
`Execution mode`	Also called *view* mode, this is the first mode to appear after you start the CLI. It is a base mode from where you can perform basic commands such as show, exit, quit, help, and enable.
Privileged execution mode	Also called *enable* mode, in this mode you can run additional basic commands such as debug, write, and show.
Configure mode	Also called *configure terminal* mode, in this mode you can run configuration commands and go into other modes such as interface, router, route map, key chain, and address family. Configure mode is single user. Only one user at a time can be in configure mode.
Interface mode	In this mode you can configure protocol-specific settings for a particular interface. Any setting you configure in this mode overrides a setting configured in router mode.
Router mode	This mode is used to configure router-specific settings for a protocol such as BGP or OSPF.

© 2026 IP Infusion Inc. Proprietary

Command Mode Tree

The diagram below shows the common command mode hierarchy.

Figure 1. Common command modes



To change modes:

1. Enter privileged executive mode by entering **`enable`** in Executive mode.
2. Enter configure mode by entering **`configure terminal`** in Privileged Executive mode.

The example below shows moving from executive mode to privileged executive mode to configure mode and finally to router mode:

```
> enable mypassword
# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
(config)# router ospf
(config-router)#
```

Note: Each protocol can have modes in addition to the common command modes. See the command reference for the respective protocol for details.

Transaction-based Command-line Interface

The OcNOS command line interface is transaction based:

- Any changes done in configure mode are stored in a separate *candidate* configuration that you can view with the `show transaction current` command.
- When a configuration is complete, apply the candidate configuration to the running configuration with the command.

- If a `commit` fails, no configuration is applied as the entire transaction is considered failed. You can continue to change the candidate configuration and then retry the `commit`.
- Discard the candidate configuration with the `abort transaction` command.
- Check the last aborted transaction with the `show transaction last-aborted` command.
- Multiple configurations cannot be removed with a single . You must remove each configuration followed by a `commit`.

Note: All commands MUST be executed only in the default CML shell (**`cmlsh`**). If you log in as root and start **`imish`**, then the system configurations will go out of sync. The **`imish`** shell is not supported and should not be started manually.

INTRODUCTION

This document describes managing OcNOS devices using NetConf.

This document is intended for network administrators and engineering professionals who configure and manage devices running OcNOS.

There are three different northbound applications (CLI, NetConf, and SNMP) in OcNOS. All the northbound applications are text-based, with each command usually associated to a specific task.

OcNOS NetConf supports transactions as described in Transactions.

© 2026 IP Infusion Inc. Proprietary

NETCONF QUICK START

The NetConf protocol defines a simple mechanism for managing a network device, retrieving configuration data information, and uploading new configuration data.

NetConf uses a simple RPC-based mechanism to communicate between a client and a server. The client can be a script or application typically running as part of a network manager. The server is usually a network device.

A NetConf session is the logical connection between a network administrator or network configuration application and a network device. A device must support at least one NetConf session and can support multiple sessions. Global configuration attributes can be changed during any session, and the effects are visible during all sessions. The candidate, running, and startup configurations are shared across all sessions.

Note: A detailed list of protocol modules supported through NetConf is in the NetConf command reference.

NetConf Clients

The OcNOS NetConf solution is tested using the OpenYuma Yangcli and netopeer CLI applications. This document uses the OpenYuma Yangcli application to demonstrate NetConf operations.

Refer to the standard Yangcli operations.

Check the release notes for the version of OcNOS you are using for where to download the IP Infusion Inc. Yang modules for NetConf clients.

Notes:

The following points are to be considered while using NetConf get-config and get operations:

- To improve the readability of the output, the subtree filter based sget-config and sget operations are used in this document.
- Yangcli's `get` and `get-config` operation time out while retrieving large amount of data. To handle this, either increase the timeout option of Yangcli, or use subtree filters used to limit the data.
- A Yangcli operation can end abruptly while fetching large amount of data. This is purely Yangcli's own system resource limitation check. To overcome this, use subtree filters or the simple SSH client.
- Yangcli is not parsing float value correctly when the value range is in between 0 to -1.

SSH client usage is explained in SSH Client (page 95).

Periodic GET Caching

To handle the `get` and `get-config` operation time out while retrieving large amount of data, there is a caching mechanism that retrieves and stores the data for easy access. A timestamp, `lastgettime`, will be saved for all the `get` operations for any path. An automatic trigger is updated at every interval and a cache is generated.

© 2026 IP Infusion Inc. Proprietary

Install Yangcli

1. Check out the git repository, using the commit e8a7bf3f2b09946880ce014fd756bcd945b0c713.
2. Apply the patch.
3. Compile and install Open Yuma components. Refer to the README for more details.

Download Yang Files

Check the release notes for the version of OcNOS you are using for where to download the IP Infusion Inc. Yang modules for NetConf clients.

Copy the files to /usr/share/yuma/modules/netconfcentral on the host machine where the client application runs. This system path only works with OpenYuma's Yangcli client application. If running a different client application, follow the respective reference document to copy the Yang files to the appropriate location.

© 2026 IP Infusion Inc. Proprietary

YANGCLI OPERATIONS

Establish Connection

These are the steps to establish a connection between the NetConf client and the server running on the device.

```
# yangcli server=<ip_address> user=<user_name> password=<password>
ip_address: Address of the device to be managed
user_name & password: User account detail for authentication
```

The interactive version of this command is shown below:

```
# yangcli yangcli> connect

Enter string value for leaf <user>
yangcli:connect> <user_name>

Enter string value for leaf <server>
yangcli:connect> <ip_address>

Enter string value for leaf <password>
yangcli:connect> <password>
```

The OcNOS NetConf server supports both IPv4 and IPv6 connections.

Notes:

In NETCONF (Network Configuration Protocol), the maximum number of sessions over SSH is indeed a configurable parameter that can be defined in the `netconfd.yang` module. Here's a generalized example of how such a parameter might be defined in a `netconfd.yang` file.

```
leaf max-sessions {
  description
    "Specifies the maximum number of sessions
     that can be accepted.";
  type uint32;
  default 1024;
}
```

- At present, the `netconfd`software is utilizing the default setting of 1024 for the `max-session` parameter. Consequently, the maximum number of concurrent NETCONF sessions is restricted to 1023 due to this default configuration.
- Attempting to create the 1024th session in the `netconfd` deletes the oldest session with session-id 1 and creates a new session with session-id 1.
- Enabling the OpenConfig Translation feature causes the NETCONF server to restart, necessitating the NETCONF client to reconnect and establish a new session. When enabling the OpenConfig Translation feature, you must specify the namespace to ensure the NETCONF server operates within the correct context.

© 2026 IP Infusion Inc. Proprietary

Configure the Device

Configuration details are placed in an XML file and sent to the `netconfd` server. You must refer to the OcNOS NetConf Command Reference to prepare the XML based configuration file with the correct hierarchy. If the hierarchy is not correct, yangcli throws an error.

One portion of the BGP protocol module Yang model is presented below. This module is a submodule for the parent OcNOS yang module.

```
submodule ipi-bgp-peer {
   yang-version 1.1;
   belongs-to  ipi-bgp { prefix ipi-bgp; }
   import feature-list {
      prefix feature-list;
      revision-date 2021-05-03;
   }
   import cml-data-types {
      prefix cml-data-types;
      revision-date 2021-05-03;
   }
   import ipi-bgp-types {
      prefix ipi-bgp-types;
      revision-date 2021-05-27;
   }
   revision "2021-06-11" {
      description "Added dependency constraint between name and direction attrs for the distribute-
list, prefix-list, filter-list and route-map CLI's";
      reference " 0.5.4.";
   }
   grouping peer-grouping {
      description
      "List of BGP neighbors configured on the local system, uniquely
                    identified by peer IPv[46] address";
      list peer {
         when " /bgp/bgp-instance/peer/config/peer-as or /bgp/bgp-instance/peer/config/mapped-peer-
group-tag ";
         key "peer-address";
         description
         "List of BGP neighbors configured on the local system, uniquely identified by peer IPv[46]
address";
         leaf peer-address  {
         type  leafref {
            path  "../config/peer-address";
         }
         description "Reference to the address of the BGP peer used as a key in the peer list";
         } // END of peer-address definition.
         list bgp-password {
            when " /bgp/bgp-instance/peer/config/peer-as or /bgp/bgp-instance/peer/config/mapped-
peer-group-tag ";
            key "password auth-key-encrypt";
            max-elements 1;
            description
            "list for BGP password";
            leaf password  {
               type  leafref {
                  path  "../config/password";
               }
            description "Use this attribute to enable authentication-key";
         } // END of password definition.
```

Based on the hierarchy in the Yang module, the following XML file named bgp.xml is created with the configuration data. The `bgp.xml` file is referenced in the `edit-config` operation specified below.

```
<bgp xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-bgp">
   <bgp-instance>
      <bgp-as>100</bgp-as>
      <config>
         <bgp-as>100</bgp-as>
      </config>
      <timers>
         <config>
            <hold-time>9</hold-time>
            <keep-alive>3</keep-alive>
         </config>
      </timers>
   </bgp-instance>
</bgp>
```

Use this command to globally set or reset the keepalive and holdtime values for all the neighbors.

```
yangcli ocnos@10.12.45.253> edit-config config=@/root/bgp.xml

Filling container /edit-config/input/target:\
RPC OK Reply 15 for session 1:
```

OcNOS supports hitless NetConf merge operation. The hitless feature blocks southbound calls to configure the provisioned configuration if it was already configured. Because of this feature, errors like “QOS already enabled” are not reported when you try to enable QoS again/repeatedly. Also, southbound calls are blocked when you try to delete/unset a non-existing configuration.

Configuration objects are merged while provisioning configuration incrementally (different/same attributes) for the same object. However this cannot be done for attributes or objects having RANGE (such as VLAN range) and MULTIPLE values (such as OSPF passive interface address) type attributes, therefore they are treated as different objects and southbound calls are allowed for every individual object. As per hitless functionality, all these objects are supposed to be merged, and a single object is expected to be sent southbound to do the configuration/un-configuration. But there will be duplicate calls southbound as was present in earlier releases. This limitation is planned to be addressed in upcoming releases.

Note: List of payloads required to configure the switch is documented in OcNOS NetConf Command Reference document.

To disable QoS, use edit-config's MERGE operation (default-operation=merge) with the below payload:

```
<qos xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-qos">
   <global>
      <config>
         <enable-qos operation="delete"/>
      </config>
   </global>
</qos>
```

Limitations

edit-config Operation

When using the `edit-config` operation with the `create` action and providing only key attributes to create a YANG list that references another YANG list, it will not result in the creation of any new configuration. This is because the referenced list already exists, and only key attributes are specified.

For instance, consider the following payload:

```
<isis xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-isis">
  <interfaces>
    <interface operation="create">
      <name>eth1</name>
      <config>
        <name>eth1</name>
      </config>
    </interface>
  </interfaces>
</isis>
```

In this example, the leaf node `/isis/interfaces/interface/config/name` serves as a `leafref` reference to `interfaces/interface/name`. Essentially, it means that the `interface` list within the `isis` module is referring to the `interface` list in the `interface` module. The provided payload contains only the key attribute for the `interface` list, indicating an intention to create a new `interface` list. However, since the referenced interface already exists, this configuration has no impact.

To make a valid and meaningful configuration change, setting other non-key attributes of the referenced interface list is necessary. For instance, the following payload is both valid and purposeful:

```
<isis xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-isis">
  <interfaces>
    <interface operation="create">
      <name>eth1</name>
      <config>
        <name>eth1</name>
        <minimal>level-1-only</minimal>
      </config>
    </interface>
  </interfaces>
</isis>
```

This payload not only specifies the key attribute but also includes additional non-key attributes for the referenced interface list, allowing for a meaningful configuration update.

Default-Operations

1. `merge` and `replace` is supported as default-operation.
2. `delete`, and `replace` is supported as operations at container and leaf level. (`operation="delete"`), (`operation="replace"`)
3. `create` is supported as operation at container level. Create operation is not supported at leaf level. (`operation="create"`).
4. `remove` is supported as operation at container and leaf level (`operation="remove"`).

Table 8. Edit-config operations

		Default operation		
		Merge	**Replace**	**None**
Create	Object	Y	Y	
	Leaf			
Delete	Object	Y	Y	
	Leaf	Y	Y	

Table 8. Edit-config operations (continued)

		Default operation		
Merge	Object	Y	Y	
	Leaf	Y	Y	
Replace	Object	Y	Y	
	Leaf	Y	Y	
Remove	Object	Y	Y	
	Leaf	Y	Y	
Legend:	Y	Supported		
	Blank	Not supported; error will be returned		

Configuration

Retrieve Candidate Configuration

Candidate configuration datastore is used to hold configuration data that can be manipulated without impacting the device's current configuration. The candidate configuration is a full configuration data set that serves as a work place for creating and manipulating configuration data. Additions, deletions, and changes can be made to this data to construct the desired configuration data.

Note: All NetConf clients share the single candidate configuration store.

```
>sget-config /bgp source=candidate
Filling list /bgp:
RPC Data Reply 1 for session 2:

rpc-reply {
   data {
      bgp {
         bgp-instance 100 {
            bgp-as 100
            config {
               bgp-as 100
            }
         timers {
            config {
               hold-time 9
               keep-alive 3
            }
         }
      }
   }
}
```

Limitations

A failed `edit-config` request results in the loss of previously successful configurations.

Use case examples:

1. Verify that no data has been sent to the candidate datastore:

```
yangcli ocnos@127.1> sget /ipi-interface:interfaces/interface/config/description
```

- RPC data reply:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <interfaces xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-interface">
      <interface>
        <name>eth2</name>
        <config>
          <name>eth2</name>
          <description>initial description</description>
        </config>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

2. Send an `edit-config` request with a valid payload:

```
yangcli ocnos@127.1> edit-config config=@/tmp/payload1.xml
```

- Verify the candidate datastore:

```
yangcli ocnos@127.1> sget-config source=candidate /ipi-
interface:interfaces/interface/config/description
```

- RPC data reply:

```
<description>payload1</description>
```

3. Send an `edit-config` request that triggers an error:

```
yangcli ocnos@127.1> edit-config config=@/tmp/payload2.xml
```

- Error reply:

```
<rpc-error>
  <error-type>application</error-type>
  <error-tag>missing-element</error-tag>
  <error-severity>error</error-severity>
  <error-message xml:lang="en">expected key leaf in list</error-message>
</rpc-error>
```

- Verify the candidate datastore after the failed request:

```
yangcli ocnos@127.1> sget-config source=candidate /ipi-
interface:interfaces/interface/config/description
```

- RPC data reply:

```
<description>initial description</description>
```

The previously successful configuration `(payload1)` is lost, and the candidate reverts to the earlier state.

The following are the payload examples:

- payload1.xml

```
<interfaces xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-interface">
  <interface>
    <name>eth2</name>
    <config>
      <name>eth2</name>
      <description>payload1</description>
    </config>
```

```
  </interface>
</interfaces>
```

- payload2.xml

```
<interfaces xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-interface">
  <interface>
    <config>
      <name>eth3</name>
      <description>payload2</description>
    </config>
  </interface>
</interfaces>
```

Commit Candidate Configuration

A <commit> operation can be performed at any time that causes the device's running configuration to be set to the value of the candidate configuration.

```
yangcli ocnos@10.12.45.253> commit

RPC OK Reply 16 for session 1
```

Retrieve Running Configuration

Configuration data is the set of writable data that is required to transform a system from its initial default state into its current state. You can use the `get-config` operation to fetch the running configuration data.

```
>sget-config /bgp source=running
Filling list /bgp:
RPC Data Reply 1 for session 2:

rpc-reply {
   data {
      bgp {
         bgp-instance 100 {
            bgp-as 100
            config {
               bgp-as 100
            }
            timers {
               config {
                  hold-time 9
                  keep-alive 3
               }
            }
         }
      }
   }
}
```

Retrieve Running Configuration and State Data

State data is the additional data on a system that is not configuration data such as read-only status information and collected statistics. You can use the `get` operation to fetch a protocol module's running configuration and state data.

```
>sget-config /bgp source=running
Filling list /bgp:
RPC Data Reply 1 for session 2:
```

© 2026 IP Infusion Inc. Proprietary

```
rpc-reply {
   data {
      bgp {
         bgp-instance 100 {
            bgp-as 100
            config {
               bgp-as 100
            }
            timers {
               config {
                  hold-time 9
                  keep-alive 3
               }
            }
         }
      }
   }
}
```

Copy Running Configuration to Startup

```
> copy-config source=running target=startup
```

The equivalent OcNOS command is:

```
# write
```

YANG Unique Datamodel Support

The key attributes in YANG datamodels are unique and it considers a combination of all its attributes as unique. For example, if the unique attributes are ‘IP’ and ‘Port’, the combination IP/Port is unique but each of these attributes can be repeated as long as it is combined with a different attribute. The check for uniqueness of the key attributes is already enforced in datamodel check and the validation of CLI / NetConf configurations is done by CML validation layer.

However, sometimes even some non-key attributes need to be unique within a list. With the unique datamodel support, the uniqueness in YANG lists with non-key attributes is also checked by a CML validation layer. The check is defined in the CML datamodel XML, validated by the CMLd. Without a validation for this uniqueness in CML layer, the commits pass on to backend modules and it failst the back-end check resulting in an unsuccessful `commit dry-run` functionality. This feature ensures that the non-key attributes which need to be unique are also checked in CML validation layer before the commit operation starts.

Example

Consider the following configuration:

```
ip vrf TEST1000
rd 173.200.0.101:1000
route-target both 0:1000

ip vrf TEST1001
rd 173.200.0.101:1000
route-target both 0:1001
```

Here, `rd`,which is not a key attribute, must be unique across the VRFs. The VRFs are listed using key of `vrf-name`, but it does not ensure that some attributes in the list entry are unique. The 'unique' keyword is used to ensure that.

Limitations

- The following IP addresses do not support unique attribute validation:
  - Secondary IP address
  - IPv6 address
- If an L3 interface is moved to a part of the member ports, the IP address configured on that member port cannot be set on the aggregated port in a single commit transaction, i.e., if the physical interface and the newly created PO interface have the same IP address, an error will occur. The configuration must be done using separate commit transactions as shown below:

```
interface eth1
 ip address 10.1.1.1/24
!
OcNOS(config)#interface po1
OcNOS(config-if)#exit
OcNOS(config)#interface eth1
OcNOS(config-if)#channel-group 1 mode active
OcNOS(config-if)#commit
OcNOS(config-if)#exit
OcNOS(config)#show running-config interface eth1
!
interface eth1
 channel-group 1 mode active
!
OcNOS(config)#show running-config interface po1
!
interface po1
!
OcNOS(config)#interface po1
OcNOS(config-if)#ip address 10.1.1.1/24
OcNOS(config-if)#commit
OcNOS(config-if)#exit
OcNOS(config)#
OcNOS(config)#show running-config interface eth1
!
interface eth1
 channel-group 1 mode active
!
OcNOS(config)#show running-config interface po1
!
interface po1
 ip address 10.1.1.1/24
!
OcNOS(config)#
```

Error Messages

NetConf operations return protocol, management layer, and protocol module errors. The example below depicts an error returned by a protocol module.

Copy the content below into `bgp_err.xml`.

```
<bgp xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-bgp">
   <bgp-instance>
      <bgp-as>200</bgp-as>
         <config>
            <bgp-as>200</bgp-as>
         </config>
         <timers>
            <config>
```

```
               <hold-time>9</hold-time>
               <keep-alive>3</keep-alive>
            </config>
         </timers>
      </bgp-instance>
</bgp>
```

Execute the following command and commit the changes:

```
yangcli tbyran@10.12.45.253> edit-config config=@/root/bgp_err.xml

Filling container /edit-config/input/target: RPC OK Reply 12 for session 1:

yangcli tbyran@10.12.45.253> commit

mgr_rpc: got invalid reply on session 1 (invalid XPath expression syntax) RPC Error
Reply 13 for session 1:

rpc-reply {
   rpc-error {
      error-type application
      error-tag operation-failed
      error-severity error
      error-app-tag general-error
      error-path /bgp/bgp-instance[bgp-as='200']/config
      error-message '%% BGP is already running, AS is 100'
      error-info {
         error-number 127
      }
   }
   rpc-error {
      error-type application
      error-tag operation-failed
      error-severity error
      error-app-tag general-error
      error-path /bgp/bgp-instance[bgp-as='200']/timers/config
      error-message '%% AS number mismatch'
      error-info {
         error-number 4294962321
      }
   }
}
```

Warning Messages

NetConf operations return protocol module warnings. The example below depicts a warning returned by a protocol module.

Copy the content below into `intf_warn.xml`

```
<interfaces xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-interface">
   <interface>
      <name>ce1/1</name>
      <config>
         <name>ce1/1</name>
         <vrf-name>management</vrf-name>
      </config>
      <ipv4 xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-if-ip">
         <config>
            <primary-ip-addr>100.12.23.3/24</primary-ip-addr>
         </config>
      </ipv4>
   </interface>
</interfaces>
```

To receive warnings, NetConf client must subscribe to receive notifications.

```
yangcli ocnos@127.1> create-subscription

RPC OK Reply 3 for session 1:
```

Execute the following command and commit the changes:

```
yangcli ocnos@127.1> edit-config config=@/home/ocnos/intf_warn.xml

Filling container /edit-config/input/target:
RPC OK Reply 4 for session 1:
yangcli ocnos@127.1> commit

RPC OK Reply 6 for session 1:


Incoming notification:
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2021-07-07T14:03:39Z</eventTime>
   <severity>info</severity>
   <eventClass>config</eventClass>
   <warning-notification xmlns="http://ipinfusion.com/ns/zebmcli">
      <warningMsg>%% IP address removed due to enabling VRF management</warningMsg>
   </warning-notification>
</notification>
```

Note: RFC 6241 is not detailed enough about how a NetConf server reports warnings. Therefore, OcNOS uses this warning reporting method.

Candidate Configuration Store Lock

Candidate configuration store is shared across all the NetConf clients. Client application must acquire lock if they want to provision the device without other client's hindrance

NetConf Client 1: locks candidate configuration store.

```
yangcli ocnos@127.1> lock target=candidate
RPC OK Reply 1 for session 4:
```

NetConf Client 2: attempts to acquire the lock now, which leads to an access-denied error.

```
yangcli ocnos@127.1> lock target=candidate
RPC Error Reply 1 for session 5:
rpc-reply {
  rpc-error {
    error-type protocol
    error-tag lock-denied
    error-severity error
    error-app-tag no-access
    error-message 'lock denied'
    error-info {
      session-id 0
      error-number 268
    }
  }
}
```

Now error “access-denied” is received by NetConf client 2 when it attempts to provision the device.

```
yangcli ocnos@127.1> edit-config config=@/home/ospf_payload.xml

Filling list /ospfv2:
RPC Error Reply 3 for session 4:

rpc-reply {
  rpc-error {
    error-type application
    error-tag in-use
    error-severity error
    error-app-tag no-access
    error-path /nc:rpc/nc:edit-config/nc:config
    error-message 'config locked'
    error-info {
      error-number 301
    }
  }
}
```

NetConf client 1 releases the lock

```
yangcli ocnos@127.1> unlock target=candidate
RPC OK Reply 2 for session 4:
```

NetConf client 2 can acquire the lock now, since no other client is holding the lock.

```
yangcli ocnos@127.1> lock target=candidate
RPC OK Reply 2 for session 5:
```

Startup configuration Store Lock

NetConf client 1 locks startup configuration store

```
yangcli ocnos@127.1> lock target=startup
RPC OK Reply 14 for session 4:
NetConf client 2 trying to modify the configuration store leads to error.
yangcli ocnos@127.1> copy-config source=running target=startup
RPC Error Reply 10 for session 5:

rpc-reply {
  rpc-error {
    error-type protocol
    error-tag in-use
    error-severity error
    error-app-tag no-access
    error-message 'config locked'
    error-info {
      error-number 301
    }
  }
}
```

Running Configuration Store Lock

Running configuration lock handling across multiple NetConf clients is not supported. But across command line interface and NetConf clients is supported. Hence this section documentation is skipped for now.

© 2026 IP Infusion Inc. Proprietary

Force Unlock

Admin user has provision to forcibly release the lock held by other users on running configuration store. This command is not supported for candidate and startup configuration stores.

NetConf client 1 locks the running configuration store

```
yangcli ocnos@127.1> lock target=running
RPC OK Reply 6 for session 6:
```

Netconf client 2 tries to acquire running configuration store leads to an error.

```
yangcli ocnos@127.1> lock target=running
RPC Error Reply 21 for session 4:

rpc-reply {
  rpc-error {
    error-type protocol
    error-tag lock-denied
    error-severity error
    error-app-tag no-access
    error-message 'lock denied'
    error-info {
      session-id 6
      error-number 268
    }
  }
}
```

NetConf client 2 (network-admin) decides to acquire the lock forcibly.

```
yangcli ocnos@127.1> force-unlock target=running

RPC OK Reply 22 for session 4:

yangcli ocnos@127.1> lock target=running

RPC OK Reply 23 for session 4:
```

Force unlock shall be issued with a timer value “after=<0-600> seconds” this command would inform existing lock holder about his lock expiry timeline.

Neconf client 2 (only network-admin) issues force-unlock command with a timer value of 60 seconds. Now after 60 seconds any other user is allowed to lock the running configuration store.

```
yangcli ocnos@127.1> force-unlock target=running after=60
RPC OK Reply 10 for session 6:
```

After 60 or more seconds, client shall issue lock command.

```
yangcli ocnos@127.1> lock target=running

RPC OK Reply 12 for session 6:
```

NetConf client 1 receives below notification about his lock expiry. To receive yangcli ocnos@127.1>

```
Incoming notification:
notification {
  eventTime 2017-09-20T19:51:09Z
  force_unlock {
    message '% Session running config store lock will be released in 60 seconds'
  }
}
```

© 2026 IP Infusion Inc. Proprietary

Sys-reload

Use this RPC to cold restart the device.

Example

```
yangcli ocnos@10.12.39.115> sys-reload
Enter boolean value for leaf <save-config>
yangcli ocnos@10.12.39.115:sys-reload>
false true
yangcli ocnos@10.12.39.115:sys-reload> true
RPC OK Reply 1 for session 4:
```

Sys-shutdown

Use this RPC to shut down the device.

Example

```
yangcli ocnos@10.12.39.115> sys-shutdown
Enter boolean value for leaf <save-config>
yangcli ocnos@10.12.39.115:sys-shutdown>
false true
yangcli ocnos@..39.115:sys-shutdown>true
```

Custom RPC Support in OcNOS

CLI-based Configuration RPC

Load configuration RPC

The `cmlsh` provides the `load-config` CLI command to merge a specified configuration file directly to the running configuration.

To perform this operation via NetConf, two RPCs are available, depending on the source of the configuration file.

load-config-filepath

The `load-config-file path` RPC is used to load a CLI-style configuration file from the local file system on the device and apply it to the running configuration. This operation follows a merge strategy, where configuration lines are processed sequentially, similar to pasting commands directly into the CLI.

RPC

```
       rpc load-config-filepath {
         description "This RPC will load the backup configuration in the source attribute via cli";
         input {
             leaf source {
                 type string;
                 description "Name of the source backup file to be restored into running-config";
             }
             leaf operation-type {
                 type enumeration {
```

```
                    enum continue-on-error {
                        description "Continue to process configuration data on error(default)";
                    }
                    enum rollback-on-error {
                        description "Initiate rollback when it receives error";
                    }
                }
                description "control the behavior of configuration changes when an error
                occurs during a transaction or commit operation";
            }
        }
        output {
            leaf show-cli-status {
                type string;
                description "Show the result of cli execution";
            }
            leaf show-failed-cli {
                type string;
                description "Show the list of cli Failed";
            }
        }
    }
```

Example

```
yangcli ocnos@127.1> load-config-filepath source=/home/config.txt

RPC OK Reply 1 for session 1:

yangcli ocnos@127.1>
```

load-config-server

The `load-config-server` RPC performs the same operation, but the configuration file is fetched from a remote server using supported protocols such as SCP or FTP.

```
rpc load-config-server {
        description "This RPC will apply the backup configuration in the source attribute";

        input {
            leaf mode {
                type enumeration {
                    enum scp {
                        description "Download files via scp";
                    }
                    enum tftp {
                        description "Download files via tftp";
                    }
                    enum ftp {
                        description "Download files via ftp";
                    }
                    enum sftp {
                        description "Download files via sftp";
                    }
                }
                description "Load-config download file mode";
            }
            leaf source {
                type string;
                description "Name of the source backup file to be restored into
                running-config Enter URL scp:[//server][/path/ filename]";            }
            leaf vrf {
                type string;
                description "VRF Name";
            }
```

```
            leaf operation-type {
                type enumeration {
                    enum continue-on-error {
                        description "Continue to process configuration data on error(default)";
                    }
                    enum rollback-on-error {
                        description "Initiate rollback when it receives error";
                    }
                }
                description "control the behavior of configuration changes
                when an error occurs during a transaction or commit operation";
            }
        }
        output {
            leaf show-cli-status {
                type string;
                description "Show the result of cli execution";
            }
            leaf show-failed-cli {
                type string;
                description "Show the list of cli Failed";
            }
        }
    }
```

Example

```
yangcli ocnos@127.1> load-config-server
source=scp://root:root123@10.16.99.116/home/2.txt mode=scp vrf=management

RPC Data Reply 1 for session 2:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <show-cli-status xmlns="http://ipinfusion.com/ns/zebmcli">Success<show-failed-cli
  xmlns="http://ipinfusion.com/ns/zebmcli"/>
</rpc-reply>
```

The `load-config` RPCs return execution results through two output fields:

- `show-cli-status`
- `show-failed-cli`

The `show-cli-status` field provides a summary of the overall status of the CLI execution, including any errors during the commit phase. This helps identify if the configuration was partially or fully applied.

The `show-failed-cli` field lists specific CLI commands that failed validation due to issues such as invalid input, incorrect value ranges, or unsupported commands. These outputs facilitate troubleshooting and correction of configuration errors during automated provisioning via NetConf.

Example

```
yangcli ocnos@127.1> load-config-server
source=scp://root:root123@10.16.99.116/home/2.txt mode=scp vrf=management

RPC Data Reply 1 for session 2:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <show-cli-status xmlns="http://ipinfusion.com/ns/zebmcli">% ' /profiles/hardware-
  profile/statistics/config/ingress-acl' % configuration not found
% ' /dns/vrfs/vrf[vrf-name='default']/config/lookup-enabled' % configuration not found</show-cli-
status>
  <show-failed-cli xmlns="http://ipinfusion.com/ns/zebmcli"/>
</rpc-reply>
```

Copy-text configuration

The `copy-text-config` RPC provides a way to replace the entire configuration on a device using a CML (Command Line Model) formatted text configuration file. It applies to both the running-config and startup-config and supports file transfers via standard protocols or local file access.

This feature is functionally similar to `copy-config` but is intended specifically for CML-based configurations (as used in CLI) rather than full XML data stores. It is especially useful for operators familiar with CLI-style configurations who want to load complete system configurations in one operation.

copy-text-config RPC

Syntax

```
    rpc copy-text-config {
        description "This RPC will replace the current config with the configuration in the source
attribute";

        input {
            leaf mode {
                type cml-data-types:cml_remote_mgmt_t;
                mandatory true;
                description "copy-text-config download file mode";
            }
            leaf target {
                type enumeration {
                    enum running-config {
                    description "running config-store";
                     }
                    enum startup-config {
                    description "startup config-store";
                     }
                }
                mandatory true;
                description "Type of config store";
            }
            leaf source {
                type string;
                mandatory true;
                description "Name of the source backup file to be restored into running-config Enter
URL
                <source>:[//server][/path/    filename]";            }
            leaf vrf {
                type string;
                description "VRF Name";
              }
            }
        output {
            leaf show-cli-status {
                type string;
                description "Show the result of cli execution";
              }
            }
        }
```

Parameter	Description
mode	Transfer protocol (scp, sftp, ftp, tftp, filepath)
source	Full path or URL of the source configuration file
target	Destination config (only running-config or startup-config)
vrf	VRF name for routing transfer traffic (required for remote URLs)

	Note: Not mandatory when using the default VRF.

Example

Using SCP with Management VRF

```
yangcli ocnos@127.1> copy-text-config
mode=scp
source=scp://user:pass@host/path/config.txt vrf=management
Enter enumeration value for leaf <target>
yangcli ocnos@127.1:copy-text-config> running-config
RPC Data Reply 1 for session 1:
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <show-cli-status xmlns="http://ipinfusion.com/ns/zebmcli">Success</show-cli-status>
</rpc-reply>
yangcli ocnos@127.1> copy-text-config mode=scp
source=scp://root:root123@10.16.99.116/home/178B.txt target=running-config vrf=management
2025 Jun 13 11:56:52.424 : 7020 : CML : ALERT : [CML_1]: Applying bulk configuration, performance may
be
impacted by high CPU usage Please consider limiting CPU usage by using the command 'cml bulk-config
limit
cpu enable'
Limiting CPU usage may impact commit time, but will preserve overall node performance
2025 Jun 13 11:56:52.431 : 7020 : CML : CRITI : Validating:  50% [|||||||||||||||||||||||||
             ]                                                              ETA: 00:12:10
RPC Data Reply 2 for session 1:
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <show-cli-status xmlns="http://ipinfusion.com/ns/zebmcli">Success</show-cli-status>
</rpc-reply>
yangcli ocnos@127.1>
```

Using SCP with Default VRF

```
yangcli ocnos@127.1> copy-text-config
mode=scp
source=scp://user:pass@host/path/config.txt
Enter enumeration value for leaf <target>
yangcli ocnos@127.1:copy-text-config> running-config
RPC Data Reply 1 for session 1:
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <show-cli-status xmlns="http://ipinfusion.com/ns/zebmcli">Success</show-cli-status>
</rpc-reply>
```

Using Local File Path

```
yangcli ocnos@127.1> copy-text-config
mode=filepath
source=/home/ocnos/config.txt
Enter enumeration value for leaf <target>
yangcli ocnos@127.1:copy-text-config> startup-config
RPC Data Reply 1 for session 1:
rpc-reply {
  show-cli-status 'System reboot is required to load this startup-config
Copy Success'
}

yangcli ocnos@127.1>
```

© 2026 IP Infusion Inc. Proprietary

Custom show RPC

custom-cmlsh-show-rpc

The `custom-cmlsh-show-rpc` retrieves outputs of CLI show commands via NetConf.

Note: The CLI command provided as input to this RPC must always start with the keyword “show” other command types are not supported.

RPC

```
rpc custom-cmlsh-show-rpc {
    description "This RPC will execute the cmlsh show command";

    input {
        leaf source {
            type string;
            description "source indicate the cli which need to be executed via cmlsh";
        }
    }

    output {
        leaf show-cli-status {
            type string;
            description "Show the result of cli execution";
        }
    }
}
```

source = Enter the show command to be executed within “ “(double quotes)

Example 1

```
show techsupport cli
yangcli root@127.1> custom-cmlsh-show-rpc source="show techsupport status"

RPC Data Reply 1 for session 1:

rpc-reply {
  show-config-file-status 'Tech-Support Command Needs To Be Executed
Tar File is generated at  and file name begins with '''
}

yangcli root@127.1> custom-cmlsh-rpc source="show techsupport bgp"

RPC Data Reply 2 for session 1:

rpc-reply {
  show-config-file-status 'This command may take a while to complete and may cause slowness in
response.
Tar File is generated at /var/log and file name begins with 'OcNOS_tech_support_bgp''
}

yangcli root@127.1> custom-cmlsh-show-rpc source="show techsupport all"

RPC Data Reply 4 for session 1:

rpc-reply {
  show-config-file-status 'Tech-support is in progress, please wait a moment...
Previous command is still in execution. Please wait....'
}
```

```
yangcli root@127.1> custom-cmlsh-show-rpc source="show techsupport status"

RPC Data Reply 5 for session 1:

rpc-reply {
  show-config-file-status 'Tech Support Command Execution Is In Progress
Protocol Id : nsm is success
Protocol Id : ripd is success
Protocol Id : ripngd is success
Protocol Id : ospfd is success
Protocol Id : ospf6d is success
Protocol Id : isisd is success
Protocol Id : hostpd is success
Protocol Id : ldpd is success
Protocol Id : rsvpd is success
Protocol Id : mribd is success
Protocol Id : pimd is success
Protocol Id : authd is success
Protocol Id : mstpd is success
Protocol Id : imi is success
Protocol Id : onmd is success
Protocol Id : oamd is success
Protocol Id : vlogd is success
Protocol Id : vrrpd is success
Protocol Id : ribd is success
Protocol Id : bgpd is success
Protocol Id : l2mribd is success
Protocol Id : lagd is success
Protocol Id : sflow is success
Protocol Id : udld is success
Protocol Id : cmld is success
Protocol Id : cmld is success
Protocol Id : pserv is success
Protocol Id : pcepd is success

/var/log/OcNOS_tech_support_bgp_28_Apr_2024_17_19_55.tar.gz_tmp
Tar File is generated at /var/log and file name begins with 'OcNOS_tech_support_bgp''
}

yangcli root@127.1>
```

Example 2

```
yangcli ocnos@127.1>

yangcli ocnos@127.1> custom-cmlsh-show-rpc source="show running-config"

RPC Data Reply 1 for session 1:

rpc-reply {
  show-cli-status '!
feature netconf-ssh vrf management
feature netconf-tls vrf management
no feature netconf-ssh
no feature netconf-tls
!
service password-encryption
!
snmp-server enable traps link linkDown
snmp-server enable traps link linkUp
!
hardware-profile statistics voq-full-color enable
hardware-profile statistics cfm-ccm enable
!
qos enable
!
tfo Disable
```

```
errdisable cause stp-bpdu-guard
no feature telnet vrf management
no feature telnet
feature ssh vrf management
no feature ssh
feature dns relay
ip dns relay
ipv6 dns relay
feature ntp vrf management
ntp enable vrf management
!
ip vrf --1234567890ABCDE__abcde
!
ip vrf 28dx_5b_36d_5b
 rd 40:40
 route-target both 40:40
!
ip vrf __0987654321--ABCDE__abcde--XY__
!
ip vrf management
 rd 51:51
 route-target both 40:40
!
ip vrf netconf_tst
 rd 3:3
 route-target both 1:1
!
ip vrf newvrf1
!
ip vrf s2_vrf
!
ip vrf site1
 rd 2.2.2.2:2
 route-target both 1:1
 route-target both 40:40
!
ip vrf snmp-vrf
!
ip vrf snmpvrf2
!
ip vrf teslnet_test
 rd 2:2
 route-target both 1:1
!
ip vrf vrf1
!
ip vrf vrf2
!
ip vrf vrf30
!
ip vrf vrf_A
!
ip vrf vrf_B
!
mac vrf newmacvrf
 rd 300:1
 route-target both evpn-auto-rt
!
mac vrf object-test3
 router-id 102.143.73.1
!
mac vrf thisismacvrf
 rd 999:1
 route-target both evpn-auto-rt
!
mac vrf vrfmac-new
 rd 1:2
 route-target both 33:33
```

```
!
router ldp
 router-id 2.2.2.2
 transport-address ipv4 2.2.2.2
!
interface ce64
!
interface ce65
!
interface ce66
!
interface ce67
!
interface ce68
!
interface ce69
!
interface ce70
!
interface ce71
!
interface eth0
 ip vrf forwarding management
 ip address dhcp
!
interface lo
 ip address 127.0.0.1/8
 ipv6 address ::1/128
!
interface lo.management
 ip vrf forwarding management
 ip address 127.0.0.1/8
 ipv6 address ::1/128
!
interface tengig64
!
interface tengig65
!
interface xe0
!
interface xe1
!
interface xe2
!
interface xe3
!
interface xe4
!
 exit
!
router ospf 1
 ospf router-id 20.20.20.20
 network 1.2.3.0/24 area 0.0.0.0
 network 2.2.2.2/32 area 0.0.0.0
 network 10.10.10.0/24 area 0.0.0.0
 network 172.22.3.0/24 area 0.0.0.0
 network 172.25.1.0/24 area 0.0.0.0
!
router bgp 100
 neighbor 1.1.1.1 remote-as 100
 neighbor 1.1.1.1 update-source 2.2.2.2
 neighbor 5.5.5.5 remote-as 100
 neighbor 5.5.5.5 update-source 2.2.2.2
 !
 address-family vpnv4 unicast
 neighbor 1.1.1.1 activate
 neighbor 5.5.5.5 activate
 exit-address-family
```

© 2026 IP Infusion Inc. Proprietary

```
 !
 address-family vpnv6 unicast
 neighbor 5.5.5.5 activate
 exit-address-family
 !
 address-family ipv4 vrf 28dx_5b_36d_5b
 redistribute connected
 exit-address-family
 !
 address-family ipv4 vrf netconf_tst
 redistribute connected
 exit-address-family
 !
 address-family ipv4 vrf site1
 redistribute connected
 exit-address-family
 !
 address-family ipv4 vrf teslnet_test
 redistribute connected
 exit-address-family
 !
 address-family ipv6 vrf 28dx_5b_36d_5b
 redistribute connected
 exit-address-family
 !
 exit
!
!
end

!
! Software version: UFI_S9600-72XC-OcNOS-SP-PLUS-6.6.1.136-MR 06/25/2025 01:57:03
!
! Last configuration change at 17:05:42 UTC Thu May 29 2025 by root'
}

yangcli ocnos@127.1>
```

Candidate Difference RPC

netconf-config-diff RPC

The `netconf-config-diff` RPC compares the candidate configuration with the running configuration and returns their differences in a structured format (XML or JSON). This allows operators or automation systems to preview configuration changes before committing them.

RPC: `netconf-config-diff`

yang RPC

```
    rpc netconf-config-diff {
        description "This RPC will comparing two NETCONF datastores (e.g., candidate vs. running) and
        display the diff-generated";

        input {
            leaf diff-format {
                type enumeration {
                    enum xml {
                        description "Output diff format is xml(default)";
                    }
                    enum json {
```

```
                        description "Output diff format is json";
                    }
                }
                description "The diff format represents the format in which differences are
displayed";
            }
        }
        output {
            leaf show-diff {
                type string;
                description "Show the result of diff";
            }
        }
    }
```

Example RPC Request - JSON Format

Example

```
JSON format:

yangcli ocnos@0> edit-config config=@/home/ospf.xml
Filling container /edit-config/input/target:
RPC OK Reply 1 for session 1:
yangcli ocnos@0> netconf-config-diff diff-format=json
RPC Data Reply 2 for session 1:
rpc-reply {
  show-diff '--- /root/.yuma/running.conf       2025-01-05 18:05:21.636195650 +0000
+++ /root/.yuma/candidate.conf  2025-01-05 18:05:21.344195650 +0000
@@ -243,6 +243,19 @@
       }
     }
   },
+  "ospfv2":{
+    "processes":{
+      "process":[
+        {
+          "ospf-id":"1",
+          "config":{
+            "ospf-id":"1",
+            "vrf-name":"default"
+          }
+        }
+      ]
+    }
+  },
   "trigger-failover":{
     "config":{
       "admin-status":"disable"'
}
```

Example RPC Request - XML Format

```
yangcli ocnos@0> netconf-config-diff diff-format=xml

RPC Data Reply 3 for session 1:
rpc-reply {
  show-diff '--- /root/.yuma/running.conf       2025-01-05 18:05:54.589195650 +0000
+++ /root/.yuma/candidate.conf  2025-01-05 18:05:54.301195650 +0000
@@ -232,6 +232,17 @@
     </error-disable>
   </global>
 </interfaces>
+<ospfv2 xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-ospf">
+  <processes>
+    <process>
```

```
+      <ospf-id>1</ospf-id>
+      <config>
+        <ospf-id>1</ospf-id>
+        <vrf-name>default</vrf-name>
+      </config>
+    </process>
+  </processes>
+</ospfv2>
 <trigger-failover xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-tfo">
   <config>
     <admin-status>disable</admin-status>'
}
yangcli ocnos@0>
```

Backup-configuration

create-backup-configuration

The `create-backup-configuration` RPC creates a backup of the current running configuration.

Syntax

```
rpc create-backup-configuration {
     description "This RPC will create a backup based on the current running-config";
 }
```

Example

```
yangcli ocnos@10.12.44.177> create-backup-configuration

RPC OK Reply 1 for session 1:

yangcli ocnos@10.12.44.177>
```

show-backup-configurations

The backup configuration can be checked with the following RPC command:

Syntax

```
rpc show-backup-configurations {
    description "This RPC will retrieve all backup file names from the system";

    output {
        leaf show-backup-configurations {
            type string;
            description "Shows available backup config file names";
        }
    }
}
```

Example

```
yangcli ocnos@10.12.44.177> show-backup-configurations

RPC Data Reply 2 for session 1:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<show-backup-configurations xmlns="http://ipinfusion.com/ns/zebmcli">Filename Created-on
```

```
OcNOS_1669930083.conf Dec-01, 2022, 21:28:03 UTC</show-backup-configurations>
</rpc-reply>
```

restore-backup-configuration

To restore a backup configuration, use the following RPC:

```
 rpc restore-backup-configuration {
     description "This RPC will restore the backedup configuration in the source attribute";

     input {
         leaf source {
             type string;
             description "Name of the source backup file to be restored into running-config";
         }
     }
 }
```

Example

```
yangcli ocnos@10.12.44.177> restore-backup-configuration source=OcNOS_1669930083.conf

RPC OK Reply 3 for session 1:
```

Feature-netconf

set-feature-netconf-status

The `set-feature-netconf-status` RPC enables or disables the NetConf feature.

```
rpc set-feature-netconf-status {
      description "Enable/Disable NetConf feature";
      input {
          leaf status {
              type enumeration {
                  enum disable {
                      value 0;
                      description "Disable netconf feature to restrict client to connect";
                  }
                  enum enable {
                      value 1;
                      description "Enable netconf feature to allow client to connect";
                  }
              }
              mandatory true;
              description "Provide input value to enable/disable feature";
          }
      }
  }
```

Example

```
yangcli ocnos@127.1> set-feature-netconf-status

Enter enumeration value for leaf <status>
yangcli ocnos@127.1:set-feature-netconf-status> disable

RPC OK Reply 5 for session 2:

yangcli ocnos@127.1> get-feature-netconf-status

RPC Data Reply 6 for session 2:
```

```
rpc-reply {
  status disable
}

yangcli ocnos@127.1>
```

get-feature-netconf-status

The `get-feature-netconf-status` RPC retrieves the current status of the NetConf feature.

```
rpc get-feature-netconf-status {
    description "This RPC will display the netconf feature status";
    output {
        leaf status {
            type enumeration {
                enum disable {
                    value 0;
                    description "Disable netconf feature to restrict client to connect";
                }
                enum enable {
                    value 1;
                    description "Enable netconf feature to allow client to connect";
                }
            }
            description "Leaf to display status of the feature";
        }
    }
}
```

Example

```
yangcli ocnos@127.1> get-feature-netconf-status

RPC Data Reply 4 for session 2:

rpc-reply {
  status enable
}

yangcli ocnos@127.1>
```

Transaction-limit

set-transaction-limit

OcNOS supports transaction-oriented configuration management, where each configuration change is treated as part of an implicit transaction initiated by edit-config operations. These transactions are finalized using either a commit or discard-changes operation, ensuring reliable and atomic updates.

By default, OcNOS supports up to 5,000 configuration changes per transaction, but this limit can be increased by the user as needed using this RPC:

```
rpc set-transaction-limit {

    description "This RPC will allow user to set number of transactions allowed per commit";
    input {
        leaf transaction-limit {
            type uint32 {
```

```
                range "0..300000";
            }
            mandatory true;
            description "Number of operation in single netconf transaction set 0 for no limit";
        }
    }
}
```

Example

```
yangcli ocnos@127.1> set-transaction-limit

Enter uint32 value for leaf <transaction-limit>
yangcli ocnos@127.1:set-transaction-limit> 70000

RPC OK Reply 7 for session 2:

yangcli ocnos@127.1>
```

show-transaction-limit

The `show-transaction-limit` RPC displays the current transaction limit per commit.

```
   rpc show-transaction-limit {
       description "This RPC will display the transaction limit per commit";
       output {
           leaf transaction-limit {
               type string;
               description " Displays number of transaction-limit";
           }
       }
   }
```

Example

```
yangcli ocnos@127.1> show-transaction-limit

RPC Data Reply 8 for session 2:

rpc-reply {
  transaction-limit 'Max-Transaction Limit is 70000'
}

yangcli ocnos@127.1>
```

Auto-config-sync

The `auto-config-sync` (enable|disable) is used to enable or disable the `auto-config-sync` after each commit.

```
  rpc auto-config-sync {
      description "This RPC will allow the user to enable or disable auto config sync";
      input {
          leaf status {
              type enumeration {
                  enum disable {
                      description "disable config auto sync";
                  }
                  enum enable {
```

```
                        description "enable config auto sync";
                    }
                }
                description "Enable or disable auto config sync. Once enabled config sync operation
                will be started whenever any
nfiguration is performed";
            }
        }
    }
```

Example

```
yangcli ocnos@127.1> auto-config-sync status=enable

RPC OK Reply 10 for session 2:

yangcli ocnos@127.1>
```

Backend API-support for Custom GET/SET RPC

This backend API-supported feature enables direct read and write operations on the custom memory pages of CMIS modules, completely bypassing the OcNOS database. The command can go directly to the protocol module without being processed by CMLD.

The following Transceivers are required:

- These two FOC FIM38950/100 and NEC OD-QD337SCLS00N transceivers are supported in this CMIS module.
- The Custom CMIS pages supported are 1Eh, 1Fh and B0h~FFh.

GET (Read) RPC

The NetConf RPC `rpc transceiver-cmis-read` instructs the Chassis Management Module Daemon (CMMd) to execute a low-level read command on the transceiver's physical memory bus (e.g., I2C/I3C), which fetches the raw data from the hardware, bypassing the OcNOS Database.

The following RPC is added to the `ipi-platform-cmis` YANG module:

```
rpc transceiver-cmis-read {
  if-feature feature-list:HAVE_CMMD;
  description "Use this RPC to make a netconf read operation that will fetch data from custom pages";
  input  {
      leaf component-name  {
          type string;
          mandatory true;
          description "Component name";
      }
      leaf bank  {
          type uint8;
          mandatory true;
          description "Bank number";
      }
      leaf page  {
          type uint8;
          mandatory true;
          description "Memory page identifier";
      }
      leaf offset  {
          type uint8;
```

```
          mandatory true;
          description "Memory page offset";
      }
      leaf size  {
          type uint16;
          mandatory true;
          description "Memory page size";
          range "1..256";
      }
  }
  output  {
      leaf result  {
          type string;
          mandatory true;
          description "";
          range "2..512";
      }
  }
}
```

To verify the module: FIM38950/100

```
yangcli ocnos@127.1> sget /components/component/transceiver/cmis-module/eeprom/state/part-number

RPC Data Reply 1 for session 1:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <components xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-platform">
       <component>
        <name>CMIS-MODULE-1</name>
        <transceiver>
          <cmis-module>
            <eeprom>
              <state>
                <part-number>FIM38950/100</part-number>
              </state>
            </eeprom>
          </cmis-module>
        </transceiver>
      </component>
    </components>
  </data>
</rpc-reply>

yangcli ocnos@127.1>
```

Enable cmm debugging

```
ocnos#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ocnos(config)#logging level cmm 7
ocnos(config)#logging console 7
ocnos(config)#debug cmm
ocnos(config)#commit
ocnos(config)#end
ocnos#
```

Send RPC command

```
yangcli ocnos@127.1> transceiver-cmis-read component-name=CMIS-MODULE-1 bank=0 page=0xc0 offset=0x80
size=16
2025 Mar 08 19:54:35.715 : ocnos : CMM : DEBUG : TESTING CUSTOM READ: ret: 0, fp-port: 1, bank: 0,
page: 192, offset: 128, size: 16, result:
00: 03 3a e2 9f 00 00 19 d7 14 fb 00 00 00 00 00 00
2025 Mar 08 19:54:35.716 : ocnos : CMM : DEBUG : buff: "033ae29f000019d714fb000000000000"

RPC Data Reply 1 for session 2:
```

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <result xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-
platform">033ae29f000019d714fb000000000000</result>
</rpc-reply>

yangcli ocnos@127.1>
```

Debug output

```
ocnos(cmm-debug)#i2cutil dump xcvr 2 ddm page 0xc0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 18 50 04 06 03 00 00 00 00 00 00 00 00 00 35 41    ?P???.........5A
10: 7c 40 f7 dd 32 00 00 00 00 00 40 00 00 00 00 00    |@??2.....@.....
20: 00 00 00 00 00 00 00 01 0d 00 00 00 00 00 00 00    .......??.......
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
50: 00 00 00 00 00 02 11 3e 81 01 0f 3e 41 11 0d 3e    .....??>???>A??>
60: 21 55 11 3e 81 01 0f 3e 41 11 0d 3e 21 55 11 46    !U?>???>A??>!U?F
70: 81 01 0f 46 41 11 00 00 00 00 00 00 00 00 00 c0    ???FA?.........?
80: 03 3a e2 9f 00 00 19 d7 14 fb 00 00 00 00 00 00    ?:??..????......
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................

ocnos(cmm-debug)#
```

SET (Write) RPC

The NetConf RPC `rpc transceiver-cmis-write` similarly instructs the Chassis Management Module Daemon (CMMd) to execute a low-level write command on the transceiver's physical memory bus (e.g., I2C/I3C), which delivers new raw hexadecimal data to the targeted CMIS pages to change its operational parameters, entirely bypassing the OcNOS database.

The following RPC is added to the `ipi-platform-cmis` YANG module:

```
rpc transceiver-cmis-write {
  if-feature feature-list:HAVE_CMMD;
  description "Use this RPC to make a netconf write operation to set custom pages data";
  input  {
      leaf component-name  {
          type string;
          mandatory true;
          description "Component name";
      }
      leaf bank  {
          type uint8;
          mandatory true;
          description "Bank number";
      }
      leaf page  {
          type uint8;
          mandatory true;
          description "Memory page identifier";
      }
      leaf offset  {
          type uint8;
          mandatory true;
          description "Memory page offset";
      }
```

```
      leaf size  {
          type uint16;
          mandatory true;
          description "Memory page size";
          range "1..256";
      }
      leaf value  {
          type string;
          mandatory true;
          description "";
          range "2..512";
      }
  }
}
```

Send RPC command

```
yangcli ocnos@127.1> transceiver-cmis-write component-name=CMIS-MODULE-1 bank=0 page=0xc0 offset=0x80
size=3 value=01A0F1
2025 Mar 08 19:54:35.715 : ocnos : CMM : DEBUG : TESTING CUSTOM READ: ret: 0, fp-port: 1, bank: 0,
page: 192, offset: 128, size: 3, result:
80: 01 a0 f1
2025 Mar 08 19:54:35.716 : ocnos : CMM : DEBUG : buff: "01a0f1"

RPC Data Reply 1 for session 2:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <result xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-platform">01a0f1</result>
</rpc-reply>

yangcli ocnos@127.1>
```

Debug output

```
ocnos#cmm-debug
ocnos(cmm-debug)#i2cutil dump port qsfp1 ddm page 0xc0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 18 50 04 06 03 00 00 00 01 00 00 00 00 00 34 98    ?P???...?.....4?
10: 7c 49 f8 08 32 00 00 00 00 00 40 00 00 00 00 00    |I??2.....@.....
20: 00 00 00 00 00 00 00 01 0d 00 00 00 00 00 00 00    ........??.......
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
50: 00 00 00 00 00 02 11 3e 81 01 0f 3e 41 11 0d 3e    .....??>???>A??>
60: 21 55 11 3e 81 01 0f 3e 41 11 0d 3e 21 55 11 46    !U?>???>A??>!U?F
70: 81 01 0f 46 41 11 00 00 00 00 00 00 00 00 00 c0    ???FA?.........?
80: 01 a0 f1 9f 00 00 19 d7 14 fb 00 00 00 00 00 00    ?:??..????......
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................

ocnos(cmm-debug)#
```

TRANSACTIONS

OcNOS supports transaction-oriented configuration management. Transactions are created implicitly by `edit-config` operations; `commit` and `discard-changes` operations close or terminate the transactions. Successive `edit-config` operations are placed in the same transaction.

Discard Changes

Discard the transaction or candidate configuration changes.

```
yangcli root@127.1> sget-config /ospfv2 source=candidate

Filling list /ospfv2:
RPC Data Reply 63 for session 2:

rpc-reply {
   data {
      ospfv2 {
         processes {
            process 20 {
               ospf-id 20
               config {
                  ospf-id 20
                  vrf-name default
               }
            }
         }
      }
   }
}

yangcli root@127.1>

yangcli root@127.1>

edit-config config=@/home/ospf_payload.xml

RPC OK Reply 64 for session 2:

yangcli root@127.1>

yangcli root@127.1> sget-config /ospfv2 source=candidate

Filling list /ospfv2:
rpc-reply {
   data {
      ospfv2 {
         processes {
            process 20 {
               ospf-id 20
               config {
                  ospf-id 20
                  vrf-name default
               }
            }
            process 25 {
               ospf-id 25
               config {
                  ospf-id 25
                  vrf-name default
```

```
               }
            }
         }
      }
   }
}


yangcli root@127.1>

yangcli root@127.1> discard-changes

RPC OK Reply 66 for session 2:

yangcli root@127.1>

yangcli root@127.1> sget-config /ospfv2 source=candidate

Filling list /ospfv2:
RPC Data Reply 67 for session 2:

rpc-reply {
   data {
      ospfv2 {
         processes {
            process 20 {
               ospf-id 20
               config {
                  ospf-id 20
                  vrf-name default
               }
            }
         }
      }
   }
}


yangcli root@127.1>
```

Commit

Commit the transaction or candidate configuration changes.

```
edit-config config=@/home/ospf_payload.xml

RPC OK Reply 17 for session 2: yangcli ocnos@10.12.45.253> commit RPC OK Reply 18 for session 2:
yangcli ocnos@10.12.45.253> sget-config /ospfv2 source=running

Filling list /ospfv2:
RPC Data Reply 19 for session 2:

rpc-reply {
   data {
      ospfv2 {
         processes {
            process 20 {
               ospf-id 20
               config {
                  ospf-id 20
                  shutdown
                  vrf-name default
               }
            }
            process 25 {
```

```
               ospf-id 25
               config {
                  ospf-id 25
                  vrf-name default
               }
            }
         }
      }
   }
}
```

Confirmed-commit

In netconf 1.1, support for confirmed-commit capability is provided.

The confirmed-commit capability indicates that the server will support the <cancel-commit> operation and the <confirmed>, <confirm-timeout>, <persist>, and <persist-id> parameters for the <commit> operation.

commit confirmed

This RPC will initiate the confirmed-commit operation. And the committed configurations will be removed if the commit is not confirmed within the default timeout period of 600 seconds. This timeout is configurable with this option [yangcli root@127.1> commit confirmed confirm-timeout=<id>]

```
yangcli ocnos@127.1> edit-config config=@/root/a.xml

Filling container /edit-config/input/target:
RPC OK Reply 1 for session 2:

yangcli ocnos@127.1> commit confirmed

RPC OK Reply 2 for session 2:

yangcli ocnos@127.1> sget-config source=running /ospfv2

Filling container /ospfv2:
RPC Data Reply 3 for session 2:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <ospfv2 xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-ospf">
      <processes>
        <process>
          <ospf-id>2</ospf-id>
          <config>
            <ospf-id>2</ospf-id>
            <vrf-name>default</vrf-name>
          </config>
        </process>
      </processes>
    </ospfv2>
  </data>
</rpc-reply>

yangcli ocnos@127.1>
```

cancel-commit

This RPC can be used to cancel the ongoing confirmed commit operation.

```
yangcli ocnos@127.1> edit-config config=@/root/a.xml

Filling container /edit-config/input/target:
RPC OK Reply 1 for session 2:

yangcli ocnos@127.1> commit confirmed

RPC OK Reply 2 for session 2:

yangcli ocnos@127.1> sget-config source=running /ospfv2

Filling container /ospfv2:
RPC Data Reply 3 for session 2:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <ospfv2 xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-ospf">
      <processes>
        <process>
          <ospf-id>2</ospf-id>
          <config>
            <ospf-id>2</ospf-id>
            <vrf-name>default</vrf-name>
          </config>
        </process>
      </processes>
    </ospfv2>
  </data>
</rpc-reply>

yangcli ocnos@127.1> cancel-commit

RPC OK Reply 4 for session 2:

yangcli ocnos@127.1> sget-config source=running /ospfv2

Filling container /ospfv2:
RPC Data Reply 5 for session 2:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data/>
</rpc-reply>

yangcli ocnos@127.1>
```

commit confirmed persist= <id>

Persist Is used to attach an identifier to the confirmed commit operation so that only with this identifier one can cancel the ongoing confirmed-commit.

```
yangcli ocnos@127.1> commit confirmed persist=100

RPC OK Reply 12 for session 2:

yangcli ocnos@127.1> !sg

yangcli ocnos@127.1> sget-config source=running /ospfv2

Filling container /ospfv2:
RPC Data Reply 13 for session 2:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <ospfv2 xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-ospf">
      <processes>
```

```
        <process>
          <ospf-id>2</ospf-id>
          <config>
            <ospf-id>2</ospf-id>
            <vrf-name>default</vrf-name>
          </config>
        </process>
      </processes>
    </ospfv2>
  </data>
</rpc-reply>

yangcli ocnos@127.1> cancel-commit

RPC Error Reply 14 for session 2:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>rpc</error-type>
    <error-tag>data-missing</error-tag>
    <error-severity>error</error-severity>
    <error-app-tag>data-incomplete</error-app-tag>
    <error-path>/cancel-commit</error-path>
    <error-message
      xmlns:xml="http://www.w3.org/XML/1998/namespace" xml:lang="en">missing parameter</error-
message>
    <error-info>
      <error-number xmlns="http://netconfcentral.org/ns/yuma-ncx">233</error-number>
    </error-info>
  </rpc-error>
</rpc-reply>

yangcli ocnos@127.1> cancel-commit persist-id=100

RPC OK Reply 15 for session 2:

yangcli ocnos@127.1> sget-config source=running /ospfv2

Filling container /ospfv2:
RPC Data Reply 16 for session 2:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data/>
</rpc-reply>

yangcli ocnos@127.1>
```

commit

Commit can be used to confirm the ongoing confirmed-commit operation.

```
yangcli ocnos@127.1> edit-config config=@/root/a.xml

Filling container /edit-config/input/target:
RPC OK Reply 32 for session 2:

yangcli ocnos@127.1> commit confirmed

RPC OK Reply 33 for session 2:

yangcli ocnos@127.1> !sg

yangcli ocnos@127.1> sget-config source=running /ospfv2

Filling container /ospfv2:
```

```
RPC Data Reply 34 for session 2:

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <ospfv2 xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-ospf">
      <processes>
        <process>
          <ospf-id>2</ospf-id>
          <config>
            <ospf-id>2</ospf-id>
            <vrf-name>default</vrf-name>
          </config>
        </process>
      </processes>
    </ospfv2>
  </data>
</rpc-reply>
```

Transaction Limit

The default transaction limit is 5000. The maximum transaction limit is 300000. Set the transaction limit to zero (0) for no limit.

Set Transaction Limit

```
<set-transaction-limit xmlns="http://ipinfusion.com/ns/zebmcli">
   <transaction-limit>6500</transaction-limit>
 </set-transaction-limit>
yangcli ocnos@0> set-transaction-limit 6500

RPC OK Reply 1 for session 3:
```

View Transaction Limit

```
yangcli ocnos@0> show-transaction-limit

RPC Data Reply 2 for session 3:

rpc-reply {
 transaction-limit 'Max-Transaction Limit is 6500'
}
```

© 2026 IP Infusion Inc. Proprietary

NETCONF MONITORING

OcNOS supports NetConf monitoring to retrieve a schema, which includes module/sub-module and it returns the requested schema. This can be achieved by `<get-schema>` operation.

Retrieve YANG Schema

To retrieve a YANG module/sub-module:

```
Yangcli yangcli root@127.1> get-schema identifier=ipi-ospf

RPC Data Reply 4 for session 1:

rpc-reply {
  data '/*
  ….
  <displays module schema here in this field>
  ….  '
}

Yangcli yangcli root@127.1> get-schema identifier=ipi-ospf-area

RPC Data Reply 4 for session 1:

rpc-reply {
  data '/*
  ….
  <displays sub-module schema here in this field>
  ….  '
}
```

Error Messages

Error is reported, when the requested schema does not exist.

```
<yangcli root@127.1> get-schema identifier=zeb

RPC Error Reply 1 for session 5:

rpc-reply {
  rpc-error {
    error-type protocol
    error-tag operation-failed
    error-severity error
    error-app-tag no-matches
    error-path /nc:rpc/ncm:get-schema/ncm:identifier
    error-message 'no matches found'
    error-info {
      error-number 365
    }
  }
}
```

© 2026 IP Infusion Inc. Proprietary

Retrieving Schema List

OcNOS supports to retrieve schema list from `/netconf-state/schemas` using the `sget` command.

```
yangcli ocnos@10.12.12.233> sget /netconf-state/schemas

Filling container /netconf-state/schemas:
RPC Data Reply 4 for session 1:

rpc-reply {
   data {
      netconf-state {
         schemas {
            schema cml-data-types 2021-05-03 ncm:yang {
                 identifier cml-data-types
                 version 2021-05-03
                 format ncm:yang
                 namespace http://www.ipinfusion.com/yang/ocnos/cml-data-types
                 location NETCONF
            }
            schema feature-list 2021-05-03 ncm:yang {
                    identifier feature-list
                    version 2021-05-03
                    format ncm:yang
                    namespace http://ipinfusion.com/ns/feature-list
                    location NETCONF
            }
            ...
            ...

            <Lists all schemas here in this space>
            ...
            ...
         }
      }
   }
}
```

URL CAPABILITIES

Overview

NetConf allows the complete OcNOS configuration to be replaced with a full Command Management Layer (CML) configuration. It also enables the backup of configurations in XML or JSON formats from all databases to a server. These functions are achieved using the Universal Resource Location (URL) capabilities identified in the following string:

```
"urn:ietf:params:netconf:capability:url:1.0?scheme=http,ftp,file, https, sftp"
```

The HTTP, HTTPS, FTP, SFTP and file schemes in the URL capabilities provide the following operations in managing the OcNOS configurations.

- copy-config
- edit-config
- delete-config

Limitations

The following operations are not supported:

- copy-config with source as URL and target as startup and running
- copy-config with source and target with a different URL.
- copy-config with source and target with the same URL (not a valid operation)
- edit-config using HTTPs and SFTP
- delete-config from HTTPs and SFTP

Prerequisites

NetConf should be configured and active.

Copy-config

Create or replace an entire configuration datastore with the contents of another complete configuration datastore. The `:url` capability modifies the `<copy-config>` operation to accept the `<url>` element as the value of the `<source>` and the `<target>` parameters. The file that the URL refers to contains the complete datastore, encoded in XML under the element `<config>` in the namespace below:

```
"urn:ietf:params:xml:ns:netconf:base:1.0"
```

Note: Yang CLI supports the URL option only in interactive mode.

© 2026 IP Infusion Inc. Proprietary

Supported Source / Target Combinations:

Source \ Target	Candidate	Running	Startup	URL
Candidate	Not Supported	Not Supported	Not Supported	Supported
Running	Supported	Not Supported	Supported	Supported
Startup	Supported	Not Supported	Not Supported	Supported
URL	Supported	Not Supported	Not Supported	Not Supported

From HTTP to the Candidate Configuration Store

Download the configuration from the HTTP server and copy it to the candidate configuration store.

```
yangcli ocnos@10.12.16.33> copy-config target=candidate

Filling container /copy-config/input/source:
Enter the number of the selected case statement:
  1: case candidate:
       leaf candidate
  2: case running:
       leaf running
  3: case startup:
       leaf startup
  4: case url:
       leaf url
  5: case config:
       container config
Enter choice number [1 - 5]:
yangcli ocnos@10.12.16.33:copy-config> 4
Enter string value for leaf <url>
yangcli ocnos@10.12.16.33:copy-config> http://10.12.12.91:10080/dashboard/docs/copy_config_
test/zebConf.xml
RPC OK Reply 10 for session 1:
Here is the example URL with basic authentication:
http://username:password@10.12.12.91:10080/dashboard/docs/copy_config_test/zebConf.xml
```

From HTTPs to the Candidate Configuration Store

The `httpscopy-config` command may be used to copy the candidate or startup configuration to the server and vice-versa.

```
yangcli ocnos@127.1> copy-config source=candidate

Filling container /copy-config/input/target:
Enter the number of the selected case statement:

  1: case candidate:
       leaf candidate
  2: case startup:
       leaf startup
  3: case url:
       leaf url

Enter choice number [1 - 3]:
yangcli ocnos@127.1:copy-config> 3

Enter string value for leaf <url>
```

© 2026 IP Infusion Inc. Proprietary

```
yangcli ocnos@127.1:copy-config> https://user:user123@10.14.105.129//test.xml

RPC OK Reply 2 for session 3:

yangcli ocnos@127.1>
  The stored configuration can be applied at a later point from the server:

yangcli ocnos@127.1> copy-config target=candidate

Filling container /copy-config/input/target:
Enter the number of the selected case statement:

  1: case candidate:
       leaf candidate
  2: case startup:
       leaf startup
  3: case url:
       leaf url

Enter choice number [1 - 3]:
yangcli ocnos@127.1:copy-config> 3

Enter string value for leaf <url>
yangcli ocnos@127.1:copy-config> https://user:user123@10.14.105.129//test.xml

RPC OK Reply 3 for session 3:

yangcli ocnos@127.1>commit
```

From FTP to the Candidate Configuration Store

Download the configuration from FTP server and copy it to the candidate configuration store.

```
yangcli ocnos@10.12.16.33> copy-config target=candidate
Filling container /copy-config/input/source:
Enter the number of the selected case statement:
  1: case candidate:
       leaf candidate
  2: case running:
       leaf running
  3: case startup:
       leaf startup
  4: case url:
       leaf url
  5: case config:
       container config
Enter choice number [1 - 5]:
yangcli ocnos@10.12.16.33:copy-config> 4
Enter string value for leaf <url>
yangcli ocnos@10.12.16.33:copy-config> ftp://10.12.12.91/config_file/zebConf.xml
RPC OK Reply 24 for session 1:
```

Here is the example URL with basic authentication:

```
ftp://username:password@10.12.12.91/config_file/zebConf.xml
```

From SFTP to the Candidate Configuration Store

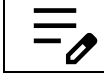
The `sftp copy-config` command may be used to copy the candidate or startup configuration to the server and vice-versa:

```
yangcli ocnos@127.1> copy-config source=candidate
```

```
Filling container /copy-config/input/target:
Enter the number of the selected case statement:

  1: case candidate:
       leaf candidate
  2: case startup:
       leaf startup
  3: case url:
       leaf url

Enter choice number [1 - 3]:
yangcli ocnos@127.1:copy-config> 3

Enter string value for leaf <url>
yangcli ocnos@127.1:copy-config> sftp://user:user123@10.14.105.129//test.xml

RPC OK Reply 2 for session 3:

yangcli ocnos@127.1>
  The stored configuration can be applied at a later point from the server:

yangcli ocnos@127.1> copy-config target=candidate

Filling container /copy-config/input/target:
Enter the number of the selected case statement:

  1: case candidate:
       leaf candidate
  2: case startup:
       leaf startup
  3: case url:
       leaf url

Enter choice number [1 - 3]:
yangcli ocnos@127.1:copy-config> 3

Enter string value for leaf <url>
yangcli ocnos@127.1:copy-config> sftp://user:user123@10.14.105.129//test.xml

RPC OK Reply 3 for session 3:

yangcli ocnos@127.1>commit
```

From a Local file to the Candidate Configuration Store

Using the file scheme support, copy the configuration from the stored local file to the candidate configuration store. Here is the example to copy local file configuration into a candidate configuration store.



Note: Local files should be stored under `/root/.yuma/` directory of the device.

```
yangcli ocnos@10.12.16.33> copy-config target=candidate
Filling container /copy-config/input/source:
Enter the number of the selected case statement:
  1: case candidate:
       leaf candidate
  2: case running:
       leaf running
  3: case startup:
       leaf startup
  4: case url:
       leaf url
```

```
  5: case config:
       container config
Enter choice number [1 - 5]:
yangcli ocnos@10.12.16.33:copy-config> 4
Enter string value for leaf <url>
yangcli ocnos@10.12.16.33:copy-config> file://zebConf.xml
RPC OK Reply 9 for session 1:
```

From the Candidate Configuration Store to HTTP

Either Running or Startup configuration can serve as the source configuration store.

Upload the configuration to the HTTP server from candidate configuration store.

```
yangcli root@127.1> copy-config source=candidate
Filling container /copy-config/input/target:
Enter the number of the selected case statement:
  1: case candidate:
       leaf candidate
  2: case startup:
       leaf startup
  3: case url:
       leaf url
Enter choice number [1 - 3]:
yangcli root@127.1:copy-config> 3
Enter string value for leaf <url>
yangcli root@127.1:copy-config> http://10.12.12.91:10080/dashboard/docs/copy_config_test/zebConf.xml
RPC OK Reply 10 for session 1:
```

Here is the example a URL with basic authentication.

```
http://username:password@10.12.12.91:10080/dashboard/docs/copy_config_test/zebConf.xml
```

From the Candidate Configuration Store to HTTPS

Upload configuration to HTTP server from candidate configuration store.

```
yangcli ocnos@127.1> copy-config source=candidate

Filling container /copy-config/input/target:
Enter the number of the selected case statement:

  1: case candidate:
       leaf candidate
  2: case startup:
       leaf startup
  3: case url:
       leaf url

Enter choice number [1 - 3]:
yangcli ocnos@127.1:copy-config> 3

Enter string value for leaf <url>
yangcli ocnos@127.1:copy-config> https://user:user123@10.14.105.129//test.xml

RPC OK Reply 2 for session 3:

yangcli ocnos@127.1>
```

From the Candidate Configuration Store to FTP

Upload configuration to FTP server from candidate configuration store.

© 2026 IP Infusion Inc. Proprietary

```
yangcli root@127.1> copy-config source=candidate
Filling container /copy-config/input/target:
Enter the number of the selected case statement:
  1: case candidate:
       leaf candidate
  2: case startup:
       leaf startup
  3: case url:
       leaf url
Enter choice number [1 - 3]:
yangcli root@127.1:copy-config> 3
Enter string value for leaf <url>
yangcli root@127.1:copy-config> ftp://10.12.12.91/config_file/zebConf.xml

RPC OK Reply 39 for session 1:
```

Here is the example URL with basic authentication.

```
ftp://username:password@10.12.12.91/config_file/zebConf.xml
```

From the Candidate Configuration Store to SFTP

Upload configuration to SFTP server from candidate configuration store.

```
yangcli ocnos@127.1> copy-config source=candidate

Filling container /copy-config/input/target:
Enter the number of the selected case statement:

  1: case candidate:
       leaf candidate
  2: case startup:
       leaf startup
  3: case url:
       leaf url

Enter choice number [1 - 3]:
yangcli ocnos@127.1:copy-config> 3

Enter string value for leaf <url>
yangcli ocnos@127.1:copy-config> sftp://user:user123@10.14.105.129//test.xml

RPC OK Reply 2 for session 3:

yangcli ocnos@127.1>
```

From the Candidate Configuration Store to the Local File

Using file scheme support, create a snapshot of the configuration store as shown in the example.

Note: Local files are created under the `/root/.yuma/` directory of the device. Storing files inside a directory is not supported.

```
yangcli root@127.1> copy-config source=candidate

Filling container /copy-config/input/target:
Enter the number of the selected case statement:

  1: case candidate:
```

```
        leaf candidate
  2: case startup:
        leaf startup
  3: case url:
        leaf url

Enter choice number [1 - 3]:
yangcli root@127.1:copy-config> 3

Enter string value for leaf <url>
yangcli root@127.1:copy-config> file://zebConf.xml

RPC OK Reply 10 for session 1:
```

Copy-config Error Messages

Following are the error messages belong to URL capability `copy-config` operation. Invalid authentication, path, IP address, port number (in case required) and XML file. Here is the example of invalid FTP authentication:

```
yangcli ocnos@10.12.16.33> copy-config target=candidate

Filling container /copy-config/input/source:
Enter the number of the selected case statement:

  1: case candidate:
        leaf candidate
  2: case running:
        leaf running
  3: case startup:
        leaf startup
  4: case url:
        leaf url
  5: case config:
        container config

Enter choice number [1 - 5]:
yangcli ocnos@10.12.16.33:copy-config> 4

Enter string value for leaf <url>
yangcli ocnos@10.12.16.33:copy-config> ftp://:admin@10.12.12.91/config_file/zebConf.xml

RPC Error Reply 32 for session 1:

rpc-reply {
  rpc-error {
    error-type rpc
    error-tag operation-failed
    error-severity error
    error-app-tag libxml2-error
    error-path /copy-config
    error-message 'xml reader start failed'
    error-info {
      error-number 212
    }
  }
}
```

Edit-config

The `<url>` element can appear instead of the `<config>` parameter. The file that the URL refers to contains the configuration data hierarchy to be modified, encoded in XML under the element `<config>` in the namespace below:

```
"urn:ietf:params:xml:ns:netconf:base:1.0"

<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <mpls xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-mpls">
      <interfaces>
         <interface>
            <name>xe3</name>
               <label-switching>
                  <config>
                     <enable/>
                  </config>
               </label-switching>
         </interface>
      </interfaces>
   </mpls>
</config>
```

Edit-config using HTTP

Provision device configuration from HTTP server using `edit-config` operation.

```
yangcli root@127.1> edit-config url=http://10.12.12.91:10080/dashboard/docs/copy_config_
test/zebConf.xml

Filling container /edit-config/input/target:
RPC OK Reply 40 for session 1:

yangcli root@127.1> sget-config source=candidate /ospfv2

Filling list /ospfv2:
RPC Data Reply 41 for session 1:

rpc-reply {
   data {
      ospfv2 {
         processes {
            process 20 {
               ospf-id 20
               config {
                  ospf-id 20
                  shutdown
                  vrf-name default
               }
            }
         }
      }
   }
}
```

Here is the example URL with basic authentication.

Edit-config using FTP

Provision device configuration from FTP server using `edit-config` operation.

```
yangcli root@127.1> edit-config url=ftp://10.12.12.91/config_file/zebConf.xml

Filling container /edit-config/input/target:
RPC OK Reply 21 for session 1:

yangcli root@127.1> sget-config source=candidate /ospfv2
```

```
Filling list /ospfv2:
RPC Data Reply 22 for session 1:

rpc-reply {
   data {
      ospfv2 {
         processes {
            process 20 {
               ospf-id 20
               config {
                  ospf-id 20
                  shutdown
                  vrf-name default
               }
            }
         }
      }
   }
}
```

Here is the example URL with basic authentication.

```
ftp://username:password@10.12.12.91/config_file/zebConf.xml
```

Edit-config using local file

Load configuration from the local storage and perform with `edit-config` operation. Load configuration from the local storage and perform with edit-config operation.

Note: Local files should be stored under `/root/.yuma/` directory of the device.

```
yangcli root@127.1> edit-config url=file://zebConf.xml

Filling container /edit-config/input/target:
RPC OK Reply 2 for session 1:

yangcli root@127.1> sget-config source=candidate /ospfv2

Filling list /ospfv2:
RPC Data Reply 3 for session 1:

rpc-reply {
   data {
      ospfv2 {
         processes {
            process 20 {
               ospf-id 20
               config {
                  ospf-id 20
                  shutdown
                  vrf-name default
               }
            }
         }
      }
   }
}
```

Edit-config Error Messages

OcNOS throws error messages for invalid authentication, invalid path, invalid IP address, and invalid port number (in case required) and invalid XML file. Here is the example for invalid FTP authentication.

```
yangcli root@127.1> edit-config url=ftp://admin:@10.12.12.91/config_file/zebConf.xml

Filling container /edit-config/input/target:
RPC Error Reply 36 for session 1:

rpc-reply {
  rpc-error {
    error-type rpc
    error-tag operation-failed
    error-severity error
    error-app-tag libxml2-error
    error-path /edit-config
    error-message 'xml reader start failed'
    error-info {
      error-number 212
    }
  }
}
```

Delete-config

Delete a configuration store. The `<url>` element can appear as the `<target>` parameter.

Delete configuration from HTTP server

Delete the configuration xml file in HTTP server from the given URL.

```
yangcli ocnos@10.12.16.33> delete-config

Filling container /delete-config/input/target:
Enter the number of the selected case statement:

  1: case startup:
       leaf startup
  2: case url:
       leaf url

Enter choice number [1 - 2]:
yangcli ocnos@10.12.16.33:delete-config> 2

Enter string value for leaf <url>
yangcli ocnos@10.12.16.33:delete-config> http://10.12.12.91:10080/dashboard/docs/copy_config_
test/zebConf.xml

RPC OK Reply 4 for session 3:
```

Here is the example URL with basic authentication.

```
http://username:password@10.12.12.91:10080/dashboard/docs/copy_config_test/zebConf.xml
```

Delete configuration from FTP

Delete the configuration file present in FTP.

```
yangcli ocnos@10.12.16.33> delete-config

Filling container /delete-config/input/target:
Enter the number of the selected case statement:

  1: case startup:
       leaf startup
  2: case url:
       leaf url

Enter choice number [1 - 2]:
yangcli ocnos@10.12.16.33:delete-config> 2

Enter string value for leaf <url>
yangcli ocnos@10.12.16.33:delete-config> ftp://10.12.12.91/config_file/zebConf.xml

RPC OK Reply 1 for session 3:
```

Here is the example URL with basic authentication.

```
ftp://username:password@10.12.12.91/config_file/zebConf.xml
```

Delete local file

Delete the configuration XML file in the local storage of the device.

Note: Local files are under `/root/.yuma/` directory of the device.

```
yangcli ocnos@10.12.16.33> delete-config

Filling container /delete-config/input/target:
Enter the number of the selected case statement:


  1: case startup:
       leaf startup
  2: case url:
       leaf url

Enter choice number [1 - 2]:
yangcli ocnos@10.12.16.33:delete-config> 2

Enter string value for leaf <url>
yangcli ocnos@10.12.16.33:delete-config> file://zebConf.xml

RPC OK Reply 3 for session 1:
```

Delete-config Error Messages

Following are the error messages related to URL capability `delete-config` operation. Invalid authentication, invalid path, invalid IP address, and invalid port number (in case required) and invalid XML file. Here is the example for invalid FTP authentication.

```
yangcli ocnos@10.12.16.33> delete-config

Filling container /delete-config/input/target:
Enter the number of the selected case statement:
  1: case startup:
       leaf startup
```

```
  2: case url:
       leaf url
```

Enter choice number [1 - 2]:

```
yangcli ocnos@10.12.16.33:delete-config> 2

Enter string value for leaf <url>
yangcli ocnos@10.12.16.33:delete-config> ftp://admin:@10.12.12.91/config_file/zebConf.xml

RPC Error Reply 9 for session 1:

rpc-reply {
  rpc-error {
    error-type protocol
    error-tag operation-failed
    error-severity error
    error-app-tag general-error
    error-path /nc:rpc/nc:delete-config/nc:target
    error-message 'operation failed'
    error-info {
      bad-value ftp://admin:@10.12.12.91/config_file/zebConf.xml
      error-number 274
    }
  }
}
```

copy-text-config

The copy-text-config provides a way to replace the entire configuration on a device using a Command Line Model (CML) formatted text configuration file. It applies to running and startup configurations and supports file transfers via standard protocols or local file access.

This feature is functionally similar to copy-config but is explicitly intended for CML-based configurations (as used in CLI) rather than full XML data stores. It is beneficial for operators familiar with CLI-style configurations who want to load complete system configurations in one operation.

With multi-VRF support added, files can now be transferred over user-defined VRFs or the management VRF. This makes it possible to source files through isolated routing domains as complex deployments require.

Feature Characteristics

- Accepts a CML-based configuration file and replaces the device's current configuration entirely.
- Supports both running-config and startup-config as targets.
- Configuration not present in the file will be removed from the device.
- Multiple warnings are issued to the user to confirm high-impact changes.
- If any failure occurs during application/validation, a rollback is automatically triggered to restore the previous state.
- Caution: Changes to management IP or authentication credentials may result in losing access to the device.

Supported Source and Target

Source Type	Target	VRF Support
URL (scp, sftp, ftp, tftp)	running-config/startup-	Yes (including user-defined

Source Type	Target	VRF Support
	config	VRFs)
Local file path	running-config/startup-config	No (local access only)

Note: Supports only running-config and startup-config.

RPC Format

```
yangcli ocnos@127.1> copy-text-config
mode    source     target             vrf
scp     /path/file running-config     management
```

Parameters

Parameter	Description
mode	Transfer protocol (scp, sftp, ftp, tftp, filepath)
source	Full path or URL of the source configuration file
target	Destination config (only running-config or startup-config)
vrf	VRF name for routing transfer traffic (required for remote URLs)

Example

```
Using SCP with Management VRF
yangcli ocnos@127.1> copy-text-config mode=scp
source=scp://user:pass@host/path/config.txt target=running-config vrf=management
```

Local File Path

```
Using SCP with Management VRF
yangcli ocnos@127.1> copy-text-config mode=scp
source=scp://user:pass@host/path/confyangcli ocnos@127.1> copy-text-config mode=filepath
source=/home/ocnos/config.txt target=startup-configig.txt target=running-config vrf=management
```

Note: Local file paths must be present on the system (For example: under `/home/ocnos/`) and do not require a VRF.

EVENT NOTIFICATION

A NetConf client registers to receive event notifications from a NetConf server by creating a subscription. The NetConf server begins notifications as events occur within the system if the subscription is successful. These event notifications will continue to be sent until either the NetConf session is terminated or the subscription terminates for some other reason.

There are four types of events supported in NetConf:

1. netconf-config-change.
2. netconf-capability-change.
3. netconf-session-start.
4. netconf-session-end.

By default the notification will not be received by the client unless the client is not subscribed.

Notifications for Operational Status

Event notifications are sent for operational status, when the following events occur:

- Service status changes from operational -> non-operational (no flow of traffic)
- Service status changes from non-operational -> operational (flow of traffic)

Note: A NetConf subscription to receive notifications must be done on the controller.

Once the NetConf session is established between the controller and leaf, the NetConf RPC `start-service-tracking` is invoked with details of the service to be tracked.

Also, a NetConf event notification and an SNMP Trap are sent when an image upgrade initiated by the controller is completed successfully.

The controller invokes NetConf RPC `sys-update-get` to download the image. The node sends the notification *sys-update-download-status* when the download finishes.

For the SNMP trap, the controller invokes NetConf RPC `start-service-tracking` with details of service to be tracked and edits the configuration to restore traffic in the node. NSM periodically checks for all the established traffic flows, using traffic rates and interfaces status. When all the required conditions match, the *send-service-tracking-status* notification is sent out.

NETCONF Notification Subscription

To subscribe to a notification as per RFC 5277, a NetConf client should send the create-subscription RPC.

Basic Subscription

To receive all notifications from the default NetConf stream:

```
yangcli ocnos@0> create-subscription

RPC OK Reply 1 for session 1:
```

Enhanced Subscription Filtering (Severity, Event Class, Stream)

CML (OCNOS) enhances the basic subscription model by supporting fine-grained filters based on following parameters:

- Severity levels (`info`, `warning`, `minor`, `major`, `critical`)
- Event classes (`config`, `state`, `message`)
- Stream name (default: NetConf)

Note:

- Filtering is supported using the `filter` and `stream` options in create-subscription.
- Replay features `<startTime>`, `<stopTime>` are not supported in create-subscription.

Event Classes and Applicable Notifications

Event Class	Applicable Notifications
config	CML_OBJECT_CREATE_NOTIF, CML_OBJECT_DELETE_NOTIF, CML_ATTRIBUTE_VALUE_CHANGE_NOTIF
state	CML_STATE_CHANGE_NOTIF
message	CML_ALARM_MESSAGE_NOTIF, CML_EVENT_MSG_NOTIF

Severity & Event Class Filter

To receive only critical state change notifications, user must provide a filter XML:

Example of XML File (`notification_filter.xml`)

```
<notification>
  <severity>critical</severity>
  <eventClass>state</eventClass>
</notification>
```

Example of Yangcli:

```
yangcli ocnos@0> create-subscription filter=@/home/xml/notification_filter.xml

RPC OK Reply 1 for session 5:
```

Stream Filter

To explicitly specify a stream, By default, the NETCONF stream is used:

Note: Currently SNMP and SYSLOG streams are not supported.

```
yangcli ocnos@0> create-subscription stream=NETCONF

RPC OK Reply 1 for session 6:
```

Example of combined Stream + Filter

User can combine both stream and filter in one subscription:

```
yangcli ocnos@0> create-subscription stream=NETCONF
filter=@/home/xml/notification_filter.xml
```

Example of RPC (with filter)

```
<netconf:rpc message-id="101"
    xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <filter netconf:type="subtree">
      <notification>
        <severity>critical</severity>
        <eventClass>state</eventClass>
      </notification>
    </filter>
  </create-subscription>
</netconf:rpc>
```

Summary

Feature	Support Details
Basic subscription	Supported
Stream filtering	Supported (NetConf only)
Severity filtering	Supported
Event class filter	Supported
Replay (start/stop)	Not supported

netconf-config-change

This notification is generated when the NetConf server detects that the <running> configuration data store has been changed by a management session. The notification summarizes the edits performed on the above mentioned data stores.

Steps to receive config change notification:

1. Create NetConf notification subscription (command: create-subscription). By default this will also subscribed for config-change notification.
2. Perform any of the following operations
    - Create
    - Merge
    - Delete

© 2026 IP Infusion Inc. Proprietary

- Replace

3. Do commit
4. Config change notification will be received.
5. Config change notification can be disabled for current session. (command: config-change-subscription status=disable)
6. Someone can check whether current session has been subscribed for config change notificaion. (command: show-config-change-subscription)

Notes:

- Config-change notification will be generated for the creation, deletion, or update of only non-leaf nodes (i.e. container, list) of YANG model.
- If a leaf node is being set or unset, indicating a modification of parent non-leaf nodes (i.e., container, list), in this case, a configuration change notification will be generated for the parent non-leaf nodes (i.e., container, list) with the operation set as 'merge'.

Example

1. Create Yangcli session
2. Subscribe for notification

```
yangcli root@0> create-subscription
   RPC OK Reply 4 for session 1
By default, this will also subscribed for config-change notification.
```

3. Someone can check whether current session has been subscribed for config change notification

```
   yangcli root@0> show-config-change-subscription
   RPC Data Reply 6 for session 1:
   <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <status xmlns="http://ipinfusion.com/ns/zebmcli">ENABLED</status>
   </rpc-reply>
```

4. Do some configuration

```
(config)#interface eth3
   (config-if)#shut
   (config-if)#commit
   Config-change notification will be received
```

5. Config change notification will be received

```
   yangcli root@0>
```

Incoming notification:

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2021-11-29T10:02:45Z</eventTime>
   <netconf-config-change xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-notifications">
   <changed-by>
   <username>root</username>
   <session-id>0</session-id>
   </changed-by>
   <datastore>running</datastore>
   <edit>
   <target>/interfaces/interface[name='eth3']/config</target>
   <operation>merge</operation>
   </edit>
   </netconf-config-change>
```

```
    </notification>
```

netconf-capability-change

This notification is generated when the NetConf server detects that the <running> or <startup> configuration data store has been changed by a management session. The notification summarizes the edits performed on the above mentioned data stores.

Note: Currently this kind of event notification is not supported.

```
notification {
   eventTime 2017-10-24T12:23:42Z
   sysCapabilityChange {
      changed-by {
         userName root
         sessionId 1
         remoteHost 127.0.0.1
   }
   added-capability http://netconfcentral.org/ns/toaster?module=toaster&revision=2009-11-20
   }
}
```

netconf-session-start

This notification is generated when a NetConf server detects client session start. Information present in this notification indicates the identity of the user.

```
notification {
   eventTime 2019-03-09T08:16:20Z
   severity info
   eventClass message
   sysSessionStart {
      sessionId 6
      remoteHost 10.12.47.71
   }
}
```

netconf-session-end

This notification is generated when a NetConf server detects client session termination. Information present in this notification indicates the identity of the user that owned the session, and why the session was terminated.

```
notification {
   eventTime 2019-03-09T08:18:55Z
   severity info
   eventClass message
   sysSessionEnd {
      userName ocnos
      sessionId 7
      remoteHost 10.12.47.71
      terminationReason closed
   }
}
```

© 2026 IP Infusion Inc. Proprietary

```
notification {
   eventTime 2019-03-09T08:19:36Z
   severity info
   eventClass message
   sysSessionEnd {
      userName ocnos
      sessionId 8
      remoteHost 10.12.47.71
      killedBy 5
      terminationReason killed
   }
}
```

Notification Cache Support in CML

The CML framework supports notification cache functionality. By default this feature is disabled, it can be enabled through configuration.

Behavior of the feature when enabled:

- When the device is reloaded or the system is upgraded, CML stores notifications in the cache.
- Notifications are stored up to max-cache-notifications (default: 100) and for a duration defined by cache-period (default: 20 minutes).
- If a NETCONF session is established and subscribes to notifications within the cache period, CML delivers all cached notifications to that session and then clears the cache.
- If no NETCONF session subscribes within the cache period, the cached notifications are discarded.

Priority-based Caching:

A special tag `<cachePriority>` can be included in data-model to override default behavior.

Example

```
<notification>
        name="bgnos-image-upgrade-done"
        desc="This belugaNOS update event is generated when the OS boots with sucess after a system
upgrade operation.">
        <type>CML_ALARM_MESSAGE_NOTIF</type>
        <subType>NB_NETCONF</subType>
        <cachePriority>HIGH</cachePriority>
        <severity>INFO</severity>
    </notification>
```

If cachePriority is set to HIGH, the notification is always stored in the cache, regardless of the max-cache-notifications limit.

Please refer the document NETCONF Command Reference for notification cache configuration.

VALIDATE CAPABILITY

Validate operation is supported only on the candidate configuration. You get a “feature not supported” error for other scenarios.

© 2026 IP Infusion Inc. Proprietary

Here is the example of successful validation.

```
yangcli root@127.1> edit-config config=@/root/config.xml default-operation=merge
Filling container /edit-config/input/target:
RPC OK Reply 3 for session 2:
yangcli root@127.1>
yangcli root@127.1> validate source=candidate
RPC OK Reply 4 for session 2:
```

WITH-DEFAULTS CAPABILITY

OcNOS supports the `with-defaults` capability which is identified by the following capability string:

```
<nc:capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-
mode=explicit&amp;also-supported=trim,report-all,report-all-tagged</nc:capability>
```

The basic mode supported by OcNOS is “explicit” mode, but you can request any other mode during a `get` or `get-config` request by adding the `with-defaults` tag as described in RFC 6243.

Explicit Mode

This is the default mode when a with-defaults tag is not specified. When data is retrieved using this mode, data nodes containing default values are reported only when explicitly set by the client.

Trim Mode

To use this mode, the client must send the `with-defaults` tag with the value “trim”:

```
<get-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <source>
    <running/>
  </source>
  <with-defaults xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">trim</with-defaults>
</get-config>
```

In this mode, the server reports only the data nodes that are not set to its default values, even if explicitly set.

Report-All Mode

To use this mode, the client must send the `with-defaults` tag with the value “report-all”:

```
<get-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <source>
    <running/>
  </source>
  <with-defaults xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">report-all</with-
defaults>
</get-config>
```

In this mode, the server does not consider any data to be default, so all data nodes are reported.

Report-All-Tagged Mode

To use this mode, the client must send the `with-defaults` tag with the value “report-all-tagged”:

```
<get-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <source>
    <running/>
  </source>
  <with-defaults xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">report-all-
tagged</with-defaults>
```

© 2026 IP Infusion Inc. Proprietary

```
</get-config>
```

This mode is similar to “report-all” mode, but the leaves that are considered the default are tagged with the ‘default’ attribute:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:wd="urn:ietf:params:xml:ns:netconf:default:1.0"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:a2be19b7-e61a
-405b-b844-59f99702316d" last-modified="2021-08-13T08:56:01Z">
  <data>
    <interfaces xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-interface">
      <interface>
        <name>eth1</name>
        <config>
          <name>eth1</name>
          <enabled wd:default="true">true</enabled>
          <switchport-status wd:default="true">false</switchport-status>
        </config>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

Edit-config Behavior

The `edit-config` command behaves according to the basic mode supported by OcNOS which is the ‘”explicit” mode. All nodes explicitly set are considered as existing so, a “create” operation on this node will fail with a “data-exists” error, otherwise, the operation succeeds. Likewise, a “delete” operation on an explicitly set node will succeed, while on a node which has its default value it receives a “data-missing” error.

As “report-all-tagged” mode is supported, the “default” attribute can also be used to set the attribute back to its default value. When this attribute is used in `edit-config`, the value of the attribute must be equal to its default value otherwise an “Invalid Value” error will be returned.

© 2026 IP Infusion Inc. Proprietary

SUPPORTED OPERATIONS

All NetConf operations are captured based on the capability. Hence, any operation falling in multiple capabilities are documented separately.

Note: Capability “base:1.0” supports candidate and running configuration store.

Table 9. Supported Operations

Capability	Operation	User Role	Supported (Yes/No)	Comments
:base:1.0	<get>	User, Operator, Engineer, Admin	Yes	
:base:1.0	<get> with subtree filter	User, Operator, Engineer, Admin	Yes	
:base:1.0	<get-config>	User, Operator, Engineer, Admin	Yes	
:base:1.0	<get-config> source=<target>	User, Operator, Engineer, Admin	Yes	
:base:1.0	<get-config> with subtree filter	User, Operator, Engineer, Admin	Yes	
:base:1.0	<edit-config> <target> as parameter	Operator, Engineer, Admin	Yes	Running configuration as target is not supported because writable-running capability is not supported; instead, candidate configuration store is supported.
:base:1.0	<edit-config> <config> as parameter	Operator, Engineer, Admin	Yes	
:base:1.0	<edit-config>: <default-operation> as merge	Operator, Engineer, Admin	Yes	
:base:1.0	<edit-config>: <delete>	Operator, Engineer, Admin	Yes	Supports attribute-level 'delete' operation only with 'merge' as default-operation. **Note:** For a leaf-level delete operation, a value is not required. If given, it is ignored by the server.

© 2026 IP Infusion Inc. Proprietary

Table 9. Supported Operations (continued)

Capability	Operation	User Role	Supported (Yes/No)	Comments
				For example: <mtu operation="delete"/> << This is enough to delete/unset a leaf. <mtu operation="delete">1560</mtu> << while processing this, the value (1560) is ignored by server. A "delete" operation at the key-leaf level is not allowed. Instead, use a delete operation at the list level (with key leaf(s) to identify list instance): <list operation="delete"> <key>value</key> </list>
:base:1.0	<edit-config>: <remove>	Operator, Engineer, Admin	Yes	Supports attribute-level 'remove' operation only with 'merge' or 'replace'. **Note:** If data is absent in the running configuration, the system will not generate a "data-missing" error for the user. Instead, it will silently ignore the "data-missing" error and continue with subsequent processing. For a leaf-level "remove" operation, a value is not required. If given, it is ignored by the server. For example:

Table 9. Supported Operations (continued)

Capability	Operation	User Role	Supported (Yes/No)	Comments
				<mtu operation=”remove”/> << This is enough to remove a leaf. <mtu operation="remove">1560</mtu> << while processing this, the value (1560) is ignored by server. A “remove” operation at the key-leaf level is not allowed. Instead, use a remove operation at the list level (with key leaf(s) to identify list instance): <list operation="remove"> <key>value</key> </list>
:base:1.0	<edit-config>: <error-option>as stop-on-error	Operator, Engineer, Admin	No	
:base:1.0	<edit-config>: <error-option>as continue-on-error	Operator, Engineer, Admin	No	
:base:1.0	<get-schema>	Operator, Engineer, Admin, User	Yes	
rollback-on-error:1.0	<edit-config>: <error-option>as rollback-on-error	Operator, Engineer, Admin	Yes	By default, this is the behavior. So there is no need to pass this error-option.
:validate: 1.1	<edit-config>: <test-option>	Operator, Engineer, Admin	No	By default, configuration entries are validated and stored, hence this operation and its parameters are not handled. External configuration store validation is not supported (i.e URL).
:base:1.0	<copy-config>: <target><source>	Engineer, Admin	Yes	<copy-config> is applicable only from running to candidate and running to startup.
:base:1.0	<lock>: < target>	Operator, Engineer, Admin	Yes	

© 2026 IP Infusion Inc. Proprietary

Table 9. Supported Operations (continued)

Capability	Operation	User Role	Supported (Yes/No)	Comments
:base:1.0	<unlock>: < target>	Operator, Engineer, Admin	Yes	
:base:1.0	<close-session> close current session	User, Operator, Engineer,Admin	Yes	
:base:1.0	<kill-session>: Close other session	User, Operator, Engineer,Admin	Yes	
:base:1.0	subtree filtering	User, Operator, Engineer, Admin	Yes	
:Startup:1.0	get-config <source=startup>	User, Operator, Engineer,Admin	Yes	
:Startup:1.0	copy-config <source=startup>	Engineer,Admin	No	Copy to candidate is not supported, and copy to running is not applicable because candidate configuration store is supported.
:Startup:1.0	copy-config <target=startup>	Engineer, Admin	Partial	Running to startup copy is supported; copying from candidate to startup is not supported.
:Startup:1.0	lock <startup>	Engineer, Admin	Yes	
:Startup:1.0	unlock <startup>	Engineer, Admin	Yes	
:Startup:1.0	validate <source=startup>	Engineer, Admin	No	Always configuration entries are validated and stored, but external configuration store validation is not supported.
:Startup:1.0	delete-config	Engineer, Admin	Yes	Running and candidate configuration store cannot be deleted.
:url:1.0	URL capability	User, Operator, Engineer, Admin	Yes	URL to startup is not supported.

© 2026 IP Infusion Inc. Proprietary

SYS-UPDATE USING NETCONF

This section contains examples of carrying out Sys-update using NetConf.

The system update feature provides the user with the option to upgrade or downgrade the OcNOS image in a router. A router's software can be upgraded when a new feature is introduced or when software bugs are fixed.

NetConf provides the user with the following options for sys-update:

1. Download the image and install.
2. To delete the downloaded image.
3. To cancel the image download which is in progress.

Note: To retain the running configuration after a system update, perform the following copy-config operation:
`copy-config source=running target=startup`

Download the Image and Install

On the Switch

Connect to `yangcli` and proceed as shown below:

`yangcli ocnos@127.1> sys-update-get url=http://10.12.40.120/jenkins-archives-3/onie/installer/OCNOS-1-3-6/EC_AS7326_56X/OCNOS_DC_IPBASE/OcNOS-1.3.6.241a-DC_IPBASE/EC_AS7326_56X-OcNOS-1.3.6.241a-DC_IPBASE-S0-P0-installer`	Download the OcNOS image. After the command, wait for some time for the image to download. Same can be verified through CLI “show installers” (Refer to validation logs)
`yangcli ocnos@127.1> sys-update-install installerName=EC_AS7326_56X-OcNOS-1.3.6.241a-DC_IPBASE-S0-P0-installer`	Install the downloaded image.

Notes:

- The `sys-update-get` CLI is updated with `known-hosts-add` parameter too support SCP and SFTP. Use the `sys-update-get known-hosts-add=true` CLI to download the OcNOS image file via SCP/SFTP.
- When `known-host-add` parameter is given as true then the IP address/hostname is added into `known_hosts` file and proceed to `sys-update`.
- If `known-host-add` parameter is not given and IP address/hostname is not present in the known_hosts file then `sys-update-get` via SCP/SFTP throws an error message `'%% Download failed, SSL peer certificate or SSH remote key was not OK'`.

Validation

Show installer output while image download is in progress:

```
tmp string is used before the image name to represent download is in progress

#show installers
/installers/tmp_EC_AS7326_56X-OcNOS-1.3.6.241a-DC_IPBASE-S0-P0-installer
#
```

Show installer output after completion of image download:

```
#show installers
/installers/EC_AS7326_56X-OcNOS-1.3.6.241a-DC_IPBASE-S0-P0-installer
```

To Delete the Downloaded Image

On the Switch

Connect to yangcli and proceed as shown below:

`yangcli ocnos@127.1> sys-update-get url=http://10.12.40.120/jenkins-archives-3/onie/installer/OCNOS-1-3-6/EC_AS7326_56X/OCNOS_DC_IPBASE/OcNOS-1.3.6.241a-DC_IPBASE/EC_AS7326_56X-OcNOS-1.3.6.241a-DC_IPBASE-S0-P0-installer`	Download the OcNOS image. After the command wait for some time for the image to get downloaded. Same can be verified through CLI “Show installers” (Refer to validation logs)
`yangcli ocnos@127.1> sys-update-delete imageName=EC_AS7326_56X-OcNOS-1.3.6.241a-DC_IPBASE-S0-P0-installer`	Delete the downloaded image.

Validation

Show installer output after completion of image download:

```
#show installers
/installers/EC_AS7326_56X-OcNOS-1.3.6.241a-DC_IPBASE-S0-P0-installer
```

Show installer output after deleting downloaded image:

```
#show installers
#
```

To Cancel the Image Download

On the Switch

Connect to yangcli and proceed as shown below:

`yangcli ocnos@127.1> sys-update-get url=http://10.12.40.120/jenkins-archives-3/onie/installer/OCNOS-1-3-6/EC_AS7326_56X/OCNOS_DC_IPBASE/OcNOS-1.3.6.241a-DC_IPBASE/EC_AS7326_56X-OcNOS-1.3.6.241a-DC_IPBASE-S0-P0-installer`	To download the OcNOS image.
`yangcli ocnos@127.1> sys-update-cancel-download`	Cancel the download.

© 2026 IP Infusion Inc. Proprietary

Validation

Show installer output while image download is in progress:

```
tmp string is used before the image name to represent download is in progress
#show installers
/installers/tmp_EC_AS7326_56X-OcNOS-1.3.6.241a-DC_IPBASE-S0-P0-installer
#
```

Show installer output after canceling of download:

```
#show installers
#
```

© 2026 IP Infusion Inc. Proprietary

SSH CLIENT

A simple SSH connection can also be used as a client application to interact with the NetConf server. Here are the steps to establish a connection and perform a get operation.

Establish a Connection

```
ssh -s ocnos@10.12.28.43 -p 830 netconf
```

Send Client Help Message to NetConf Server

Copy and paste this message in the session and perform operations without the Enter key:

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <capabilities>
   <capability>urn:ietf:params:netconf:base:1.0</capability>
 </capabilities>
</hello>]]>]]>
```

Note: Only base 1.0 capability is used here though server supports both base 1.0 and 1.1 capabilities. Because the later one mandates the XML encoding type "chunked framing" (RFC 6242, section 4.1), which is not user friendly.

Perform the `get` operation:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"/>
</rpc>]]>]]>
```

Perform `get-config` operation:

```
<rpc message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <source>
      <running/>
    </source>
  </get-config>
</rpc>]]>]]>
```

Perform `edit-config` operation:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <candidate/>
      </target>
      <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ospfv2 xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-ospf">
            <processes>
```

© 2026 IP Infusion Inc. Proprietary

```
               <process>
                  <ospf-id>20</ospf-id>
                  <config>
                     <ospf-id>20</ospf-id>
                     <shutdown/>
                     <vrf-name>default</vrf-name>
                  </config>
               </process>
            </processes>
         </ospfv2>
      </config>
   </edit-config>
</rpc>]]>]]>
```

Perform `commit` operation:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit/>
</rpc>]]>]]>
```

Perform `get-schema` operation:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <netconf-state xmlns=
      "urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
        <schemas/>
      </netconf-state>
    </filter>
  </get>
</rpc>]]>]]>
```

Perform `copy-config` operation:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <copy-config>
    <target>
       <url>file://ZebOS.conf</url>
    </target>
    <source>
      <running/>
    </source>
   </copy-config>
 </rpc>]]>]]>
```

NETCONF-SSH over User Defined VRF

OcNOS now supports netconf-ssh feature over user defined VRFs along with default and management VRFs. With this, user must be able to enable netconf-ssh feature over multiple user defined VRFs simultaneously and access the device through user defined VRF networks from remote client.

User must be able to customize the netconf ssh ports to non-default ports (other than 830).

Server Configuration for User Defined VRFs

`#configure terminal`	Enter configure mode

(config)#ip vrf vrf_test	Configure User defined VRF
(config)#feature netconf-ssh vrf vrf_test	Enable netconf-ssh feature over user defined VRF say VRF name vrf_test
(config)#commit	Commit configuration
(config)#no feature netconf-ssh vrf vrf_test	Disable netconf-ssh feature over user define VRF, and define VRF name vrf_test
(config)#commit	Commit configuration

Server Configuration for User Defined VRFs to Configure SSH Ports

#configure terminal	Enter configure mode
(config)#no feature netconf-ssh vrf vrf_test	Disable netconf-ssh feature over user defined VRF say VRF name “vrf_test”
(config)#commit	Commit configuration
(config)#netconf server ssh-port 65535 vrf vrf_test	Configure ssh port over user defined VRF.
(config)#commit	Commit configuration
(config)#feature netconf-ssh vrf vrf_test	Enable netconf-ssh feature over user defined VRF and define VRF name vrf_test to reflect the port configuration
(config)#commit	Commit configuration

Validation

```
#show netconf server
VRF Management
        Netconf SSH Server: Enabled
        SSH-Netconf Port : 830
VRF Default
        Netconf SSH Server: Enabled
        SSH-Netconf Port : 830
VRF vrf_test
        Netconf SSH Server: Enabled
        SSH-Netconf Port : 65535
```

© 2026 IP Infusion Inc. Proprietary

OPENDAYLIGHT CONTROLLER

The Opendaylight-0.12.2 user manual can be found here:

https://docs.opendaylight.org/projects/netconf/en/stable-magnesium/user-guide.html

Installation

Before installing OpenDaylight, install required JAVA modules. OpenDaylight Magnesium requires a Java 11 environment. Any version other than Magnesium can work under Java 8 environment.

```
$ sudo apt-get -y install openjdk-8-jre
$ sudo apt-get -y install openjdk-11-jre
```

OpenDaylight can be downloaded from the link below.

https://docs.opendaylight.org/en/latest/downloads.html

Unzip the tar.gz file with this command:

```
$tar xvzf opendaylight.tar.gz
```

To start OpenDaylight, navigate to the OpenDaylight folder and run following command.

```
$./bin/karaf
```

Below mentioned features are needed to be installed in the karaf shell of OpenDaylight.

```
feature:install odl-netconf-connector-all odl-netconf-topology odl-netconf-console odl-restconf odl-
mdsal-apidocs
```

Connecting with an OcNOS Device

To establish a connection between OpenDaylight and an OcNOS VM, below attached file will be used. Make sure to update host IP address and credentials in ocnos_connect.xml file depending on the NETCONF device.

Node-id for the connection can be of the user choice.

```
ocnos_connect.xml:
<node xmlns="urn:TBD:params:xml:ns:yang:network-topology">
<node-id>ocnos-3</node-id>
<host xmlns="urn:opendaylight:netconf-node-topology">x.x.x.x</host>
<port xmlns="urn:opendaylight:netconf-node-topology">830</port>
<username xmlns="urn:opendaylight:netconf-node-topology">ocnos</username>
<password xmlns="urn:opendaylight:netconf-node-topology">ocnos</password>
<tcp-only xmlns="urn:opendaylight:netconf-node-topology">false</tcp-only>
<keepalive-delay xmlns="urn:opendaylight:netconf-node-topology">10</keepalive-delay>
</node>
```

Use the following command to make connection. Make sure about the update of data in the `ocnos_connect.xml` file. “ocnos-3” in URL is node-id.

```
curl --user "admin":"admin" -H "Content-type: application/xml" -X PUT http://
localhost:8181/restconf/config/network-topology:network-topology/topology/topology-
/node/ocnos-3 -d '@ocnos_connect.xml'
```

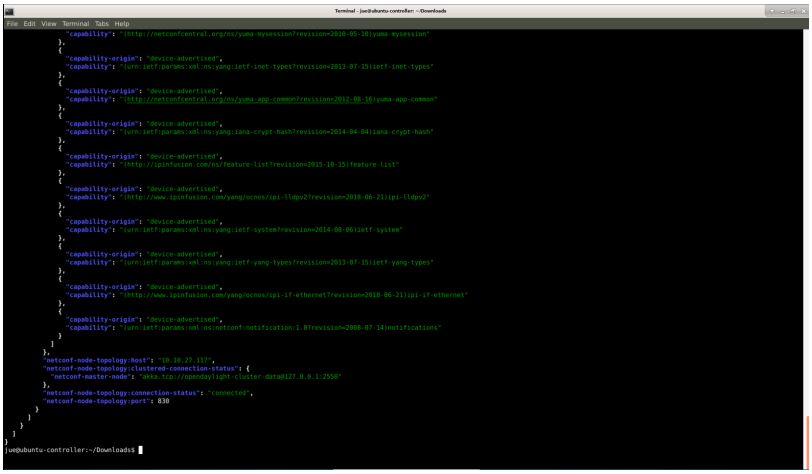
After execution of the above command, check the connection status with the following command.

© 2026 IP Infusion Inc. Proprietary

```
curl --user "admin":"admin" -H "Content-type: application/xml" -X GET http://
localhost:8181/restconf/operational/network-topology:network-topology/topology/topology-
netconf/ | jq .
```

The URL from the command can be used in a browser to access as well. ("username:password" = "admin:admin" if asked for sign-in when accessing URL.)

Results of above command:



If the status shows 'connecting', it indicates capability exchange is still ongoing and it will soon move to 'connected'.

Restconf Endpoints

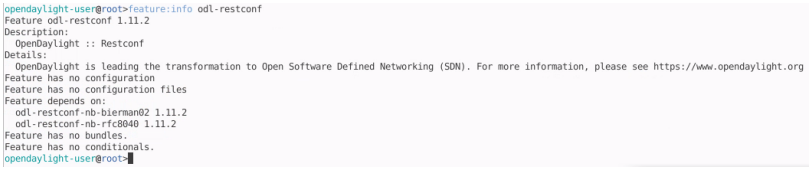
There are 2 different endpoints related to RestConf protocols:

- Draft-bierman-netconf-restconf-02, http://localhost:8181/restconf/…
- RFC 8040, http://localhost:8181/rests/…

For example:

```
GET http://localhost:8181/rests/data/network-topology:network-topology?content=config
for configuration datastore and GET http://localhost:8181/rests/data/network-topology:network-
topology?content=nonconfig for operational datastore.
```

Make sure to install all necessary features. If the odl-restconf feature is already installed, it provides both end points together.



RFC 8040 can be installed independently:

```
$ feature:install odl-restconf-nb-rfc8040
```

Note: For connecting to OcNOS, the RFC 8040 Restconf endpoint needs to be used and installed. This has support for the 'PATCH' operation which translates to Netconf 'Merge'.

Patch or Merge Operation

In the case of RFC 8040, resources for configuration and operational datastores start /rests/data/.

For example:

- `GET http://localhost:8181/rests/data/network-topology:network-topology` with response of both datastores. It is allowed to use query parameters to distinguish between them.
- `GET http://localhost:8181/rests/data/network-topology:network-topology?content=config` for configuration datastore
- `GET http://localhost:8181/rests/data/network-topology:network-topology?content=nonconfig` for operational datastore.

Also in the case of RFC 8040, if a data node in the path expression is a YANG leaf-list or list node, the path segment has to be constructed by having leaf-list or list node name, followed by an “=” character, then followed by the leaf-list or list value. Any reserved characters must be percent-encoded.

For example:

`GET http://localhost:8181/rests/data/network-topology:network-topology/topology=topology-netconf?content=config`

Retrieves data from configuration datastore for topology-netconf value of topology list is equivalent to the deprecated request.

A patch request can be used to modify an existing configuration. Currently, only yang-patch (RFC 8072) is supported. The URL would be the same as the above PUT examples.

Using JSON for the body, the headers needed for the request would be:

```
Accept: application/yang.patch-status+json
content-Type: application/yang.patch+json
```

(Change the header-type for XML accordingly.)

JSON payload

```
content-Type: application/yang.patch+json
{
  "ietf-restconf:yang-patch" : {
    "patch-id" : "0",
    "edit" : [
      {
        "edit-id" : "edit1",
        "operation" : "merge",
        "target" : "/",    //target value is the module to be configured
        "value" : {


         ### configuration goes here ###


        }
      }
    ]
  }
}
```

XML payload

```
content-Type: application/yang.patch+xml
```

```
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
    <patch-id>0</patch-id>
    <edit>
        <edit-id>edit1</edit-id>
        <operation>merge</operation>
        <target>/   </target> //target value is the module to be configured
        <value>


            ### config goes here ###


        </value>
    </edit>
</yang-patch>
```

This example shows a patch operation to edit ip-address for interface using ODL:

```
PATCH
@
http://localhost:8181/rests/data/network-topology:network-topology/topology=topology-
netconf/node=123a/yang-ext:mount/ipi-interface:interfaces/interface=eth3/ipi-if-ip:ipv4
```

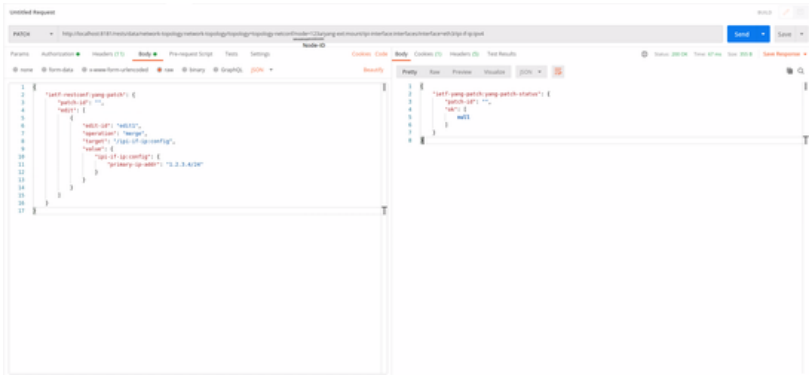
JSON payload

```
content-Type: application/yang.patch+json
{
    "ietf-restconf:yang-patch": {
        "patch-id": "",
        "edit": [
            {
                "edit-id": "edit1",
                "operation": "merge",
                "target": "/ipi-if-ip:config",
                "value": {
                    "ipi-if-ip:config": {
                        "primary-ip-addr": "x.x.x.x/24"
                    }
                }
            }
        ]
    }
}
```

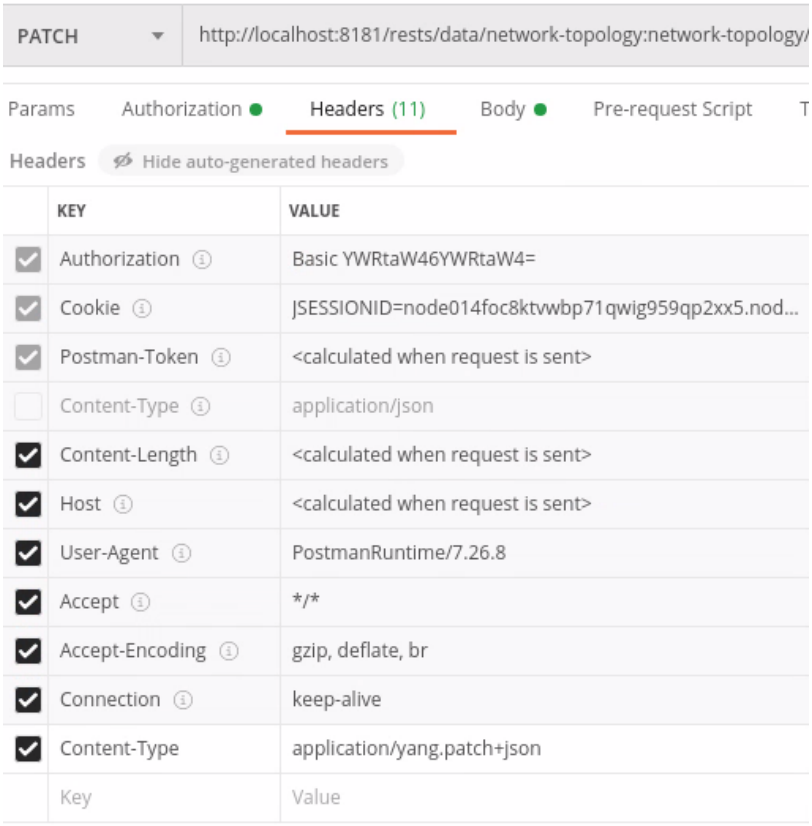
XML payload

```
content-Type: application/yang.patch+xml
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
    <patch-id>0</patch-id>
    <edit>
        <edit-id>edit1</edit-id>
        <operation>merge</operation>
        <target>/ipi-if-ip:config</target>
        <value>
            <config xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-if-ip">
                <primary-ip-addr>x.x.x.x/24</primary-ip-addr>
            </config>
        </value>
    </edit>
</yang-patch>
```

© 2026 IP Infusion Inc. Proprietary



Header for the action:



POSTMAN

Installation

```
$ sudo snap install postman
```

Postman app can be found under applications->Development->postman

For more details for downloading and installing POSTMAN desktop version:

https://gist.github.com/invinciblycool/ecc1c6e32b581b68932ac7452f4c911c

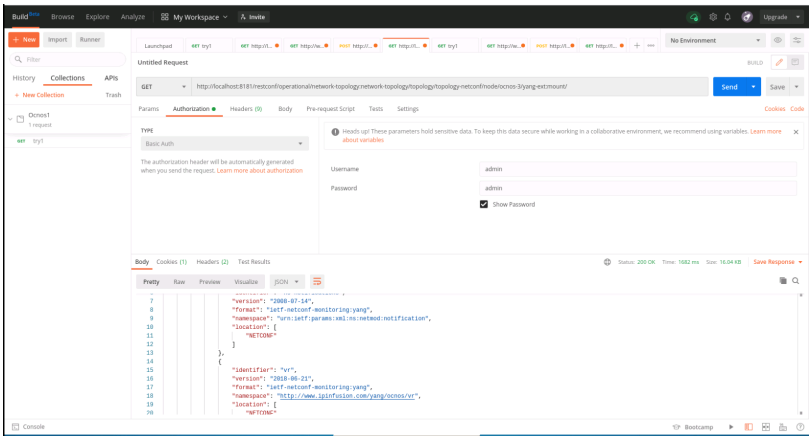
The POSTMAN desktop version should be running in the system while the web version is accessed at this link:

https://web.postman.co/

In the configuration of OpenDaylight in the POSTMAN tool, authentication should be set to basic authentication with username and password as 'admin' (as shown in the snippet above).

All the RESTCONF operations can be performed using POSTMAN tool.

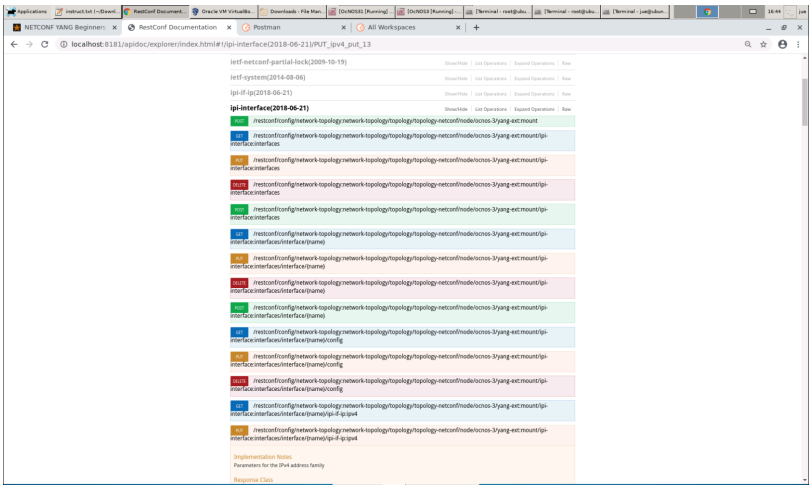
Configuration using POSTMAN is shown later in the section "edit-config".



Edit-config

Edit-config using API-DOCS

Under the mounted resources of API-DOCS, all the APIs show the possible RESTCON operations which can be performed on the node. In the snippet below, different RESTCONF operations are displayed under ipi-interface API.



Edit-config using Terminal

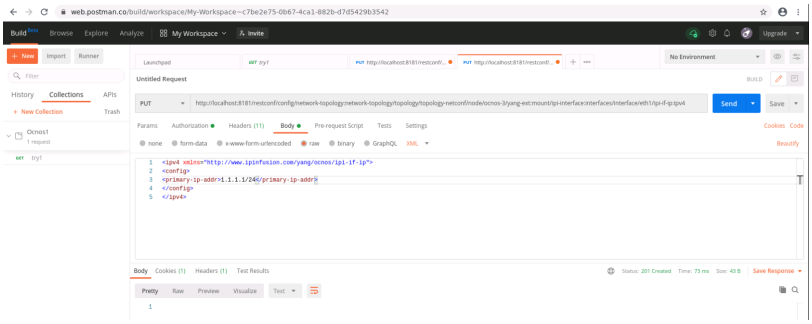
```
$ curl -user "admin":"admin" -H "Content-type: application/xml" -X PUT http://
localhost:8181/restconf/config/network-topology:network-topology/topology/topology-
netconf/node/ocnos3/yang-ext:mount/ipi-interface:interfaces/interface/eth1/ipi-if-ip:ipv4 -d
'@interface_config.xml'
```

The interface_config.xml file contents are:

```
interface_config.xml:
<ipv4 xmlns="https://www.ipinfusion.com/yang/ocnos/ipi-if-ip">
<config>
<primary-ip-addr>1.1.1.1/24</primary-ip-addr>
</config>
</ipv4>
```

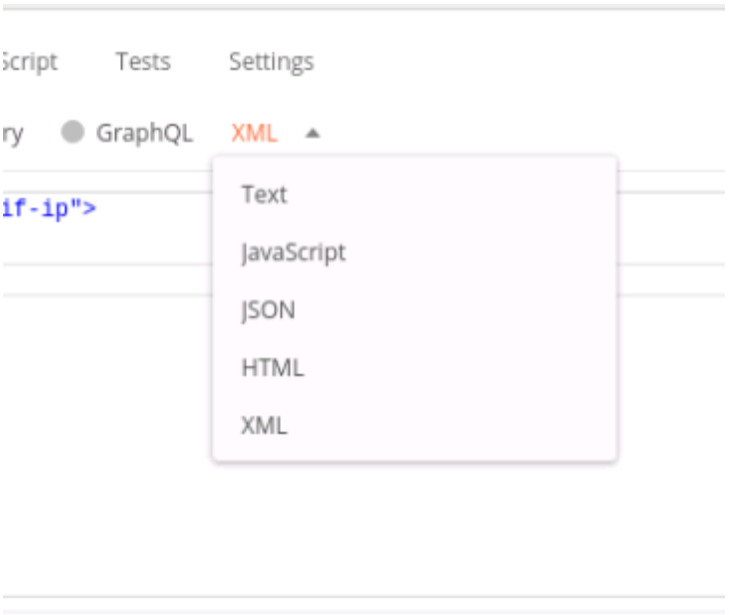
Edit-config using POSTMAN

Similar to edit-config in NetConf, RESTCONF PUT method is sent by the client to create or replace the target data resource. RESTCONF PUT operation is shown in below attached snippet. Authentication should be selected as 'basic authentication' as shown before.



In the snippet above, the URL of interface eth1 of node-is = ocnos-3 is set to receive configuration file.

The body of the request contains the configuration data. POSTMAN supports different formats for data configuration as well.



As shown in the snippet above, after sending this request for the configuration, it returns a "201 created" http response. This means that target datastore has created the data resource as per the request.

Known Issues and Troubleshooting

Mounted resources on API-Doc showing 500:internal server error

ODL API-Doc requires additional patch to enable mounted resources. This is a known issue in ODL. To apply the patch/solution for ODL,

Replace `yang-model-util-4.0.13.jar` residing at:

```
opendaylight-0.12.2/system/org/opendaylight/yangtools/yang-model-util/4.0.13/
```

with the provided JAR file.

Note: Contact IP Infusion Inc. support for the patch file.

To apply the PATCH using .JAR files

OpenDaylight shows certain parsing issues with IP Infusion Inc. generated yang modules. This in turn affects OpenDaylight functions. It can be avoided by changing certain .JAR files in ODL system.

If you have revised JAR file, these revised .JAR files will replace OpenDaylight source JAR files.

To enable the change of the PATCH, OpenDaylight features are needed to be installed again. That can be achieved by:

1. Delete the OpenDaylight/data directory.
2. Replace provided .JAR files with current OpenDaylight/system .jar files.
3. Re-run after clearing OpenDaylight/data, this ODL will have no "data" and will be affected with the PATCH.
4. Re-install all necessary karaf features.

Unavailable-capabilities in Hello Message

After connecting with netconf device, it advertises all its yang capabilities in its hello message with ODL.

If OpenDaylight shows unavailable capabilities, please try to delete all yang files residing at:

```
opendaylight-0.12.2/cache/schema/
```

and attempt to re-connect with netconf-device.

JAVA_HOME Not Set

If the `JAVA_HOME not set` error appears when executing the `$./bin/karaf` command, it can be solved by setting `JAVA_HOME` to the directory of the local JDK.

To perform that operation, you need to provide local directory address to `JAVA_HOME`.

To check where the Java binary resides:

```
$ sudo update-alternatives --config java
```

There is only one alternative in link group java (providing /usr/bin/java):

```
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
```

Nothing to configure.

In our case, the binary for Java 8 resides at:

```
/usr/lib/jvm/java-8-openjdk-amd/jre/bin/java
```

With the path known, apply this command to update BASHRC file:

```
$ echo 'export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre' >> ~/.bashrc
```

(Make sure that `JAVA_HOME` ends with `/jre`.)

OcNOS Device Compatibility

OcNOS devices with different product versions (such as SP, DC, or RON) cannot be connected simultaneously with the same ODL host machine.

To connect a device with a different OcNOS product version (when it was previously connected with another version), you need to clean the ODL schema cache.

OcNOS devices with different release versions of the same product *can* connect simultaneously to the same ODL host machine.

© 2026 IP Infusion Inc. Proprietary

NETCONF OVER TRANSPORT LAYER SECURITY

Transport Layer Security (TLS) is a cryptographic protocol that uses mutual certificate-based authentication to provide a secure and reliable connection between two devices. It is the successor to the Secure Sockets Layer (SSL) protocol.

When a NetConf session is established over TLS, the OcNOS NetConfF server acts as the TLS server, and the NETCONF client acts as the TLS client.

NetConf sessions over TLS provide several advantages over sessions that use SSH:

- SSH authenticates a client using credentials (username/password) or keys.
- TLS uses X.509 certificates to mutually authenticate the client and the server.
- Certificates can carry additional information about the client and can be centrally issued and managed (for example, by a corporate PKI).

While NETCONF sessions over SSH work well for manually managing individual devices, NETCONF over TLS is typically preferred for secure device-to-device communication and for large-scale automation.

TLS subsystem logs are integrated with the system logger (syslog) and appear (along with other OcNOS logs) in `/var/log/message` with the tag `TLS_SUBSYS`.

By default, the OcNOS NETCONF server listens for NETCONF over TLS connections on TCP port 6513.

Note: A maximum of 3 NETCONF sessions over TLS are supported.

Topology

This topology depicts OcNOS device acts as NETCONF server over TLS.

NETCONF client (for example, netopeer2-cli running in a Docker container) connects over TCP port 6513 using mutual TLS.

The same TLS server certificate on OcNOS can be used both for NETCONF over TLS and for Streaming Telemetry over TLS.

Figure 2. Netconf over TLS Topology

Generating TLS Certificates

For OcNOS 7.0 and later, use a single CA certificate to generate both the server and client certificates, so that NETCONF over TLS and Streaming Telemetry can use the same TLS infrastructure.

© 2026 IP Infusion Inc. Proprietary

This procedure is common for both NETCONF and Telemetry.

Generate CA Certificates

1. Generate a CA certificate key on a TLS client (Linux host, not on OcNOS):

```
openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 -keyout rootCAKey.pem -out
rootCACert.pem
```

   This generates:

   - `rootCAKey.pem` – CA private key
   - `rootCACert.pem` – CA certificate

   Use this same CA certificate (`rootCACert.pem`) to sign both server and client certificates.

Generate Server Certificates

2. On OcNOS, generate the server key and certificate signing request (CSR) from the cmlsh prompt:

```
#enable
         #crypto pki generate rsa common-name ipv4 <server-ip>
```

   where `<server-ip>` is the IP address that NETCONF/TLS and Telemetry clients use to reach the device.

```
#crypto pki generate rsa common-name ipv4 10.12.93.111
```

3. Display the CSR on OcNOS:

```
# show crypto csr
-----BEGIN CERTIFICATE REQUEST-----
MIICXDCCAUQCAQAwFzEVMBMGA1UEAwwMMTAuMTIuOTMuMTExMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEA7dTLZpS7sTCrFdP4gQD6X4+PHtYKK7jQdSaM
WaX20dkPqxN8/6VHuY3+fE13eeUSPrDbdSOzPq6f/Rdd/xG76Qi95yYKZnUkTz1/
sF50LfdxL2u+Xg25TA4+Lh6EHYxbmLwX6/o4b8E+78CH+NftH6g1/2YN14PC3lI1
Jlar+YEAzlyy8q1RAHbiazyJmgR0n4KxdtU/c/nzU8zP4OWtJdNquAAMkQUcWwBg
0a88FpFEm5SkQRslLX2BdofpO0SCiMJVkYRJH8yirxKwHx7JBg7EtDovofY585rj
zBZBTycsWYrpWGh+YUm+5ZbEHN+NC2r9UqvLVuHyYWRPB40jxQIDAQABoAAwDQYJ
KoZIhvcNAQELBQADggEBAD0/4MxtfJrz3jeBAMUwIjTBFauTI2rJ0AMxjIcPfvel
wLi/nK4HBU3Ucg9yUqfPYwfi3gLa901AJT5UVPZV0C783nNBrR9GRa91rL0+0Ksa
hptlTCvFug/N7oEIADQ2IHskNTyCSW7MeJaz06amhiGHp+QNjNulAmsXUjPOzKsJ
OILOmBvD10N+GvtvBMhJrxDKzCda9auSdk1If1BRdfmNttnfthoK3PWK5kmkyhlr
l4mz3Mvd9E5UvsMfL8u72FibwaHa3b862/YQdbidN2LS8xjCZTxeitbJooLzywhb
hR99BENUG8zTbmIa8BD2Nt8F3mlo/31kD8eO2QTLqfI=
-----END CERTIFICATE REQUEST-----
#
```

Copy the complete CSR output and save it on the Linux host as `ServerCert.csr`.

4. On the Linux host, create a SAN extension file `san.ext` to unify the server certificate for NETCONF and Telemetry:

```
cat > san.ext << 'EOF'
subjectAltName = @alt_names
[alt_names]
DNS.1 = <server-ip>
DNS.2 = OcNOS
EOF
```

5. Use the CSR and SAN extension file to generate the server certificate:

```
openssl x509 -req -sha256 -in ServerCert.csr -extfile san.ext -CA rootCACert.pem -CAkey rootCAKey.pem
-CAcreateserial -out ServerCert.pem -days 365
```

© 2026 IP Infusion Inc. Proprietary

This command creates the server certificate file which is used by NETCONF over TLS server and Streaming Telemetry server (gNMI TLS):

- `ServerCert.pem` – TLS server certificate for OcNOS

Generate Client Certificates

Use the same CA certificate (`rootCACert.pem`) for all client certificates.

6. Generate client certificate request configuration file `ClientCertReq.config`:

```
cat > ClientCertReq.config << 'EOF'
            [req]
            distinguished_name = dn
            prompt = no
            [dn]
            CN = <common-name>
            C  = IN
            L  = BNG
            O  = IPI
         EOF
```

where `<common-name>` can be the client's IP address or any identifier.

For multiple clients, generate separate client certificates, but always sign them with the same CA (`rootCACert.pem`).

```
[req]
               distinguished_name = dn
               prompt = no

               [dn]
               CN = 10.12.65.10 ---------- >   <Apply TLS client IP here>
               C = IN
               L = BNG
               O = IPI
               OU = IPI-QA
```

7. Generate the client key and CSR:

```
openssl req -newkey rsa:2048 -keyout ClientKey.pem -out ClientCert.csr -config
./ClientCertReq.config -nodes -days 365
```

This command creates these two files

- `ClientKey.pem` – client private key
- `ClientCert.csr` – client certificate

8. Sign the client CSR using the CA:

```
openssl x509 -req -sha256 -in ClientCert.csr -out ClientCert.pem -CA rootCACert.pem -CAkey
rootCAKey.pem -CAcreateserial -days 365
```

This command creates the `ClientCert.pem` – client certificate file.

9. On the Linux host, manually import (SCP) signed server certificate and CA-root certificate to OcNOS (`/usr/local/etc/tls/certs`).
   - File should be in PEM format for CA-root certificate and named `ca.pem` (`/usr/local/etc/tls/certs/ca.pem`).
   - OcNOS server certificate file should be in PEM format and named `cert.pem` (`/usr/local/etc/tls/certs/cert.pem`).

© 2026 IP Infusion Inc. Proprietary

```
cp ServerCert.pem cert.pem
            cp rootCACert.pem ca.pem

            scp cert.pem ocnos@<mgmtIP>:/tmp
            scp ca.pem   ocnos@<mgmtIP>:/tmp

            Note: Under OcNOS privileged executive mode move .pem files to
/cfg/usr/local/etc/tls/certs/

         mv cert.pem ca.pem /cfg/usr/local/etc/tls/certs/
```

The same `cert.pem` and `ca.pem` are used by both NETCONF over TLS and Telemetry TLS.

Enabling NETCONF Server over SSH and TLS

This section describes how to enable the NETCONF server on OcNOS using either SSH or TLS.

1. Enter configuration mode and enable NETCONF over SSH:

```
# Enter configuration mode
               OcNOS#configure terminal
               OcNOS(config)#feature netconf-ssh
               OcNOS(config)#netconf-ssh port 830
               OcNOS(config)#commit


               # show netconf server
               VRF Management
               Netconf SSH Server: Enabled
               SSH-Netconf Port : 830
            ...
```

Note: To use the user defined ssh port number, make sure the port value is within the range 1024 to 65535.

2. Enable NETCONF over TLS. Before enabling NETCONF over TLS, ensure that `cert.pem` and `ca.pem` have been copied to `/cfg/usr/local/etc/tls/certs/` path.

```
(config)#feature netconf-tls
               (config)#netconf server tls-port 6513
               (config)#commit


               # show netconf server
               VRF Management
               Netconf TLS Server: Enabled
               TLS-Netconf Port : 6513
               VRF Default
            Netconf Server: Disabled
```

Note: Before enabling TLS, you must generate or import PKI certificates using crypto pki commands.

Installing Netopeer2

Perform the following steps to install netopeer docker image.

1. On a Linux host, start Netopeer2 Docker Container.

```
sudo apt install docker.io
           sudo docker load -i netopeer_docker_image.tar.gz
         sudo docker run -itd --name netopeer -h netopeer --network=host netopeer_minideb
```

2. To run netopeer2-cli inside the container:

```
sudo docker exec -it netopeer netopeer2-cli
```

3. Copy Client Certificates created in Generating TLS Certificates (page 107) to the container:

```
CONTAINER=netopeer   # container name
           docker exec $CONTAINER mkdir -p /root/certs
           docker cp ClientCert.pem  "$CONTAINER":/root/certs
           docker cp ClientKey.pem   "$CONTAINER":/root/certs
         docker cp rootCACert.pem  "$CONTAINER":/root/certs
```

Creating a new Docker Image (Netopeer2)

Perform the following steps to create docker image for Netopeer:

1. Create a docker file named `netopeer_dockerfile` with the following code text.

```
FROM bitnami/minideb:bullseye

# Install netopeer2 dependencies
RUN install_packages git cmake build-essential bison flex libpcre2-dev
libev-dev libavl-dev libprotobuf-c-dev protobuf-c-compiler swig python3-dev
lua5.2 pkg-config libpcre++-dev openssl libssl-dev libcrypto++-dev zlib1g-dev gdb
RUN install_packages libssh-dev
RUN install_packages apt-transport-https ca-certificates
RUN update-ca-certificates

RUN mkdir NetConfServer
WORKDIR /NetConfServer

# Download GitHub repos
RUN git clone https://github.com/CESNET/libyang.git
RUN git clone https://github.com/sysrepo/sysrepo.git
RUN git clone https://github.com/CESNET/libnetconf2.git
RUN git clone https://github.com/CESNET/netopeer2.git

# ARG netopeer_version=v2.1.62
# ARG libyang_version=v2.1.80
# ARG libnetconf2_version=v2.1.34
# ARG sysrepo_version=v2.2.73

# ARG netopeer_version=v2.1.59
# ARG libyang_version=v2.1.55
# ARG libnetconf2_version=v2.1.31
# ARG sysrepo_version=v2.2.60

# ARG netopeer_version=v2.1.49
# ARG libyang_version=v2.1.30
# ARG libnetconf2_version=v2.1.28
# ARG sysrepo_version=v2.2.36

# These are the working versions
ARG netopeer_version=v2.1.42
ARG libyang_version=v2.1.4
ARG libnetconf_version=v2.1.25
ARG sysrepo_version=v2.2.12

# Install libyang
WORKDIR /NetConfServer/libyang
```

```
# Error starts between v2.1.17 and v2.1.18
# RUN git checkout -f a415e6031b3ab14c7b2e3e600fc73c6d387a1712
RUN git checkout -f $libyang_version
RUN mkdir build
WORKDIR /NetConfServer/libyang/build
RUN cmake ..
RUN make
RUN make install
RUN ldconfig

# Install sysrepo
WORKDIR /NetConfServer/sysrepo
RUN git checkout -f $sysrepo_version
RUN mkdir build
WORKDIR /NetConfServer/sysrepo/build
RUN cmake ..
RUN make
RUN make install
RUN ldconfig

# Install libnetconf2
WORKDIR /NetConfServer/libnetconf2
RUN git checkout -f $libnetconf_version
RUN mkdir build
WORKDIR /NetConfServer/libnetconf2/build
RUN cmake ..
RUN make
RUN make install
RUN ldconfig

# Install Netopeer2
WORKDIR /NetConfServer/netopeer2
RUN git checkout -f $netopeer_version

# Uncomment notifications container
RUN sed -i 's/\/\*container/container/' modules/notifications@2008-07-14.yang
RUN sed -i 's/}\*\//}/' modules/notifications@2008-07-14.yang
#####

RUN mkdir build
WORKDIR /NetConfServer/netopeer2/build
RUN cmake ..
RUN make
RUN make install
```

2. Build the docker image. Execute the following command from the directory where `netopeer_dockerfile` exists.

```
docker build --no-cache -f netopeer_dockerfile -t netopeer_minideb .
```

3. Stop and remove the old container (if it exists).

```
docker stop netopeer_minideb
docker rm netopeer_minideb
```

Using Netopeer2 as NETCONF Client (Over TLS)

`netopeer2-cli` is one example of a NETCONF client that can be used to establish a NETCONF over TLS session with OcNOS. Other compliant NETCONF clients with TLS support can also be used.

1. Connect to OcNOS over TLS from netopeer2-cli prompt.

```
> connect --tls \
--host <OcNOS-IP> \
--port 6513 \
```

```
--cert /root/certs/ClientCert.pem \
--key /root/certs/ClientKey.pem \
--trusted /root/certs/rootCACert.pem
```

On success, perform standard NETCONF operations.

Examples:

To retrieve the running configuration:

```
> get-config --source running
```

To retrieve state data:

```
> get
```

If operations take too long, specify a custom RPC timeout (example):

```
> edit-config --target candidate --config=/root/ospfc1.xml --rpc-timeout=120
> commit --rpc-timeout=120
```

NETCONF-TLS over User Defined VRFs

From Release 6.5.3 OcNOS supports the netconf-tls feature over user-defined VRFs, in addition to the default and management VRFs. This allows users to enable the netconf-tls feature across multiple user-defined VRFs simultaneously, providing access to the device via user-defined VRF networks from a remote client.

Users can also customize the netconf-tls ports for user-defined VRFs, using non-default ports (other than 6513).

The process for generating client and server certificates and accessing the device through netconf-tls remains the same as described in previous sections

Server Configuration for User Defined VRFs

`#configure terminal`	Enter configure mode
`(config)#ip vrf vrf_test`	Configure User defined VRF
`(config)#feature netconf-tls vrf vrf_test`	Enable netconf-tls feature over user defined VRF say VRF name “vrf_test”
`(config)#commit`	Commit configuration
`(config)#no feature netconf-tls vrf vrf_test`	Disable netconf-tls feature over user define VRF say VRF name “vrf_test”
`(config)#commit`	Commit configuration

Server Configuration for User Defined VRFs to Configure TLS Ports

`#configure terminal`	Enter configure mode
`(config)#no feature netconf-tls vrf vrf_test`	Disable netconf-tls feature over user defined VRF say vrf name “vrf_test”

© 2026 IP Infusion Inc. Proprietary

`(config)#commit`	Commit configuration
`(config)#netconf server tls-port 65535 vrf vrf_test`	Configure TLS port over user defined VRF.
`(config)#commit`	Commit configuration
`(config)#feature netconf-tls vrf vrf_test`	Enable netconf-tls feature over user defined VRF say vrf name “vrf_test” to reflect the port configuration
`(config)#commit`	Commit configuration

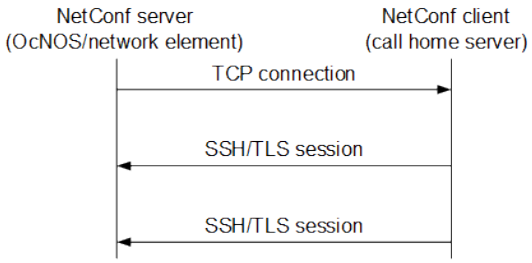
Validation

```
#show netconf server
VRF Management
        Netconf TLS Server: Enabled
        TLS-Netconf Port : 6513
VRF Default
        Netconf TLS Server: Enabled
        TLS-Netconf Port : 6513
VRF vrf_test
        Netconf TLS Server: Enabled
        TLS-Netconf Port : 65535
```

CALL HOME

By default, in the NetConf protocol (RFC 6241), a NetConf client application initiates the connection towards the NetConf server in the network element (OcNOS device). However, for certain use cases, such as in the presence of firewalls or NAT, it is useful to have “call home” functionality where the connection process is reversed, and the NetConf server initiates the connection to the NetConf client. As shown in Figure 3, this process is standardized by IETF in RFC 8071.

Figure 3. RFC 8071 NetConf call home functionality



OcNOS supports call home feature (only for SSH) at the NetConf server side. You can use any standard NetConf client application which supports call home functionality. (Call home support in the NetConf client application [Yangcli] is not supported.)

Call home is generally useful for initial deployment and ongoing management of networking elements.

Note: Call Home allows a maximum of 5 Call Home servers, with a single NETCONF session per server.

Configuration

To configure the call home server and other required metadata, use the `ipi-management-server` module. The Yang tree below lists the related attributes.

```
module: ipi-management-server
  +--rw netconf-server
     +--rw callhome!
     |  +--rw feature-enabled    empty
     |  +--rw management-port?   string
     |  +--rw netconf-client* [name]
     |  |  +--rw name       string
     |  |  +--rw address    string
     |  |  +--rw port?      inet:port-number
     |  +--rw reconnect!
     |     +--rw enable                empty
     |     +--rw retry-max-attempts?   uint8
     |     +--rw retry-interval?       uint32
     +--rw debug
        +--rw callhome-debug?   empty
```

For details, see the *NetConf Command Reference*.

NETCONF ACCESS CONTROL MODEL USER GUIDE

Overview

The NETCONF Access Control Model (NACM) provides a standardized framework for managing user access and permissions within the NETCONF environment. It defines how access to configuration and operational data is controlled, ensuring that only authorized users or groups can view, modify, or execute specific operations on the device.

NACM enables administrators to define fine-grained permissions for different users through both rule-based and group-based access control. It governs which RPCs and configuration data can be viewed or modified. It supports multiple rule types that applies to modules, protocol operations, data nodes, and notifications to offer flexible and precise policy enforcement.

By applying NACM, network devices can be managed more securely and consistently. It helps prevent unauthorized configuration changes, ensures compliance with organizational policies, and aligns with Internet Engineering Task Force(IETF) security standards for NETCONF protocol.

Feature Characteristics

- NACM manage the roles specific permission access to read, write, and execute operation in network devices.
- **User and Group Management:**
  - **ROOT**: The Root user is a super user with unrestricted access.
  - **admin/ocnos User**: admin/ocnos users belongs to **PRIV1 group**, which has all the permission. They can create group, add users to the group, and configure NACM rules for those groups.
- **Restricted Operations:** Only PRIV1 group users (admin/ocnos) and root user can perform `copy-config` and `delete-config` operations.
- **Configuration Persistance:**To ensure NACM configurations are retained across reboots, admin and ocnos users must perform the `<copy-config>` operation with `source=running` and `target=startup`.
- **Recovery:**
  - The super user **root** with unrestricted access and is not bound by NACM rules. Any NetConf session established with the root user is considered a recovery session. During recovery, the root user can **create**, **delete**, or **update** one or more NACM rules to bring the device back to a stable state.
  - The admin/ocnos users belong to the PRIV1 group, which has full access permissions through a NACM rule that grants complete privileges to this group. During recovery, the admin/ocnos user can also create, delete, or update one or more NACM rules to restore system stability, provided the PRIV1 rule itself is not deleted.
  - The Root, admin and ocnos users can execute the `delete-configtarget=startup` operation to restore the startup configuration to its default state during recovery scenarios.
- Implemented as a YANG module (`ietf-netconf-acm`) and works with NetConf servers to dynamically enforce access controls.

© 2026 IP Infusion Inc. Proprietary

- **Rule-Based Access Control** access is controlled based on the following rule components:
  - Target: The rule applies to which specifies the data nodes, RPCs or notifications.
  - Action: Defines the access to permit or deny.
  - User/Group: Identifies the entity to which the rule applies.

Role-to-Permission Mapping in NACM

Role/User	Group	Permissions
Root	None	Full unrestricted access to all NetConf operations and configurations.
admin/ocnos	PRIV1	Full access including privileged operations like `copy-config`, `delete-config`. admin and ocnos users belongs to PRIV1 group.
Other Users	Custom	Access defined by group-specific NACM rules (for example: read-only, limited RPCs).

Configuration

Initial NACM Configuration

When the NetConf server starts, it is equipped with a default NACM configuration if no prior existing configuration. The initial configuration is as follows:

- By default, the NACM feature is disabled. To enable this feature as per the requirements, only the Root user, admin or ocnos user can send an **`edit-config`** request to set the configuration **`/nacm/enable-nacm=true`**.
- Once enabled, if no prior configuration exists under the **`/nacm`** subtree, the server will deny **read**, **write**, and **execute** access to all operations and data, except for users in the PRIV1 group **(admin/ocnos)** and the Super user **root**.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
    <enable-nacm>false</enable-nacm>
    <read-default>deny</read-default>
    <write-default>deny</write-default>
    <exec-default>deny</exec-default>
    <enable-external-groups>false</enable-external-groups>
    <groups>
      <group>
        <name>PRIV1</name>
        <user-name>admin</user-name>
        <user-name>ocnos</user-name>
      </group>
    </groups>
    <rule-list>
      <name>admin-rules</name>
      <group>PRIV1</group>
      <rule>
        <name>permit-all</name>
        <action>permit</action>
        <comment>Permit everything for PRIV1 group</comment>
      </rule>
    </rule-list>
  </nacm>
</config>
```

NetConf NACM vs CLI Role-Based Access

Category	NETCONF NACM	CLI
Role Management	Manages roles and permissions independently.	Manages roles and permissions independently.
Role Definitions	Roles defined via YANG model (ietf-netconf-acm).	Use its own role structure
Access Control Enforcement	Applies rules to NetConf operations and data nodes.	Applies rules to CLI commands.
Authentication Methods	Uses NetConf-specific authentication mechanisms.	CLI uses separate authentication mechanisms.

NACM Rules types

- **Module Rules**: These govern access control to all definitions within the YANG module.

Example: Allow read access to the ipi-interface module.

- **Operation Rules (RPC rule)**: These rules restrict access to specific protocol RPC operations or YANG actions. They are defined by the module and the RPC identifier.

Example: Deny access to `<edit-config>` for non-admin users.

- **Notification Rules**: These manages access to specific notification event type, scoped by module and notification name.

Example: Allow access to “interface-link-state-change-notification” notification for operators.

- **Data Rules**: These rules provide fine-grained access control over configuration and operational data via XPath expressions.

Example: Grant read-only access to **`/interfaces/interface[name='eth0']`**.

Benefits

- Fine-grained control over configuration and operational data.
- Prevents unauthorized changes or sensitive data exposure.

Prerequisites

- The NetConf client should include the NACM capability `urn:ietf:params:xml:ns:yang:ietf-netconf-acm` in its `<hello>` message to use the NACM feature.
- NACM must be enabled in the server configuration.
- User accounts and their corresponding group memberships should be configured before applying NACM rules, as access control is based on user and group identities.

Common NACM Rule Fields

Field	Description
`rule-name`	Unique name of the NACM rule. This is a required

	identifier and must be unique within the list of rules.
`module-name`	The name of the YANG module where the target node, RPC, action, or notification resides (e.g., ietf-interfaces, ietf-netconf). * in `module-name` allows rules to apply for all modules. This is default value.
`access-operations`	Specifies the NetConf operation types this rule applies to. Can be a space-separated list of any combination of: • create – Create a node • read – Read data (get, get-config, notification) • update – Modify existing config • delete – Remove a node • exec – Execute an RPC or action Special value: • * – Match all operations.
`action`	The decision NACM will take when this rule matches. • `permit` – Allow access. • `deny` – Deny access.
`path`	XPath expression identifying the data node(s) this rule applies to. Optional.
`rpc-name`	Name of the RPC or action (e.g., `edit-config`, `get`, or custom RPCs). Used only for `exec` operations. Optional. * – Match all rpc-names.
`notification-name`	Name of the notification node this rule applies to Optional. * – Match all notification-names.
`comment`	Optional comment/description for human readability.
`user-name`	Username to which this rule applies.
`group-name`	NACM group to which this rule applies.

Note: Only one of `path`, `rpc-name`, or `notification-name` can be specified in a rule. They are mutually exclusive and depend on the rule type:

- Use `path` for data nodes
- Use `rpc-name` for RPCs or actions
- Use `notification-name` for notifications

© 2026 IP Infusion Inc. Proprietary

Valid Path Notes for NETCONF NACM <path> Rules

When defining NACM `<path>` rules in NetConf, it is critical to use fully qualified and absolute **XPath expressions** that accurately represent the data model defined in your YANG modules. Follow the best practices below:

General Guidelines

- The `<path>` must be an absolute XPath, i.e., it must start with /.
- Use fully qualified XPath with correct namespace prefixes and declarations.
- Ensure that all prefixes and namespace URIs match those defined in the corresponding YANG modules.
- Always include prefixes for all keywords including keys inside predicates.

Example 1: Path from Single YANG Module

YANG Module: `openconfig-platform`

```
module openconfig-platform {
  namespace "http://openconfig.net/yang/platform";
  prefix "oc-platform";
}
```

Valid NACM Rule:

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>grp_several-rule-list</name>
    <group>grp_several</group>
    <rule>
      <name>grp_several-rule-1</name>
      <path xmlns:oc-platform="http://openconfig.net/yang/platform">
        /oc-platform:components/oc-platform:component
      </path>
      <access-operations>read</access-operations>
      <action>permit</action>
      <comment>grp_several-rule-1-addition</comment>
    </rule>
  </rule-list>
</nacm>
```

Example 2: Path with Augmentation Across Multiple Modules

YANG Modules:

```
module openconfig-terminal-device {
  namespace "http://openconfig.net/yang/terminal-device";
  prefix "oc-opt-term";
}
```

Valid NACM Rule:

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>grp_several-rule-list</name>
    <group>grp_several</group>
    <rule>
      <name>grp_several-rule-2</name>
      <path xmlns:oc-platform="http://openconfig.net/yang/platform"
            xmlns:oc-opt-term="http://openconfig.net/yang/terminal-device">
        /oc-platform:components/oc-platform:component/oc-opt-term:optical-channel/oc-opt-
term:config/oc-opt-term:frequency
      </path>
      <access-operations>read</access-operations>
      <action>permit</action>
      <comment>grp_several-rule-2-addition</comment>
    </rule>
```

```
  </rule-list>
</nacm>
```

Example 3: Path with Keys in Predicates (Prefix Required)

Keys must include the appropriate prefix, to describe the function of the key.

Valid XPath Example:

```
<path xmlns:oc-platform="http://openconfig.net/yang/platform"
      xmlns:oc-opt-term="http://openconfig.net/yang/terminal-device">
  /oc-platform:components/oc-platform:component[oc-platform:name='OCH-0/1']/oc-opt-term:optical-
channel
</path>
```

Creating NetConf RPC for NACM

RPC Configurations for NACM

Edit-config RPC for enabling NACM

The `edit-config` RPC is used to enable NACM by updating the relevant configuration in the YANG module.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <enable-nacm>true</enable-nacm>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

Edit-config RPCs for group

An RPC is used to **create the user group "PRIV2"** and add the user test to it in the YANG module.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <groups>
          <group>
            <name>PRIV2</name>
            <user-name>test</user-name>
          </group>
        </groups>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

NACM RPCs for Module Rule

An `edit-config` RPC is used to **permit read access** to the **ipi-interface** module for the user group **PRIV2** by configuring rule in the YANG module.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>PermitReadInterfaces</name>
            <module-name>ipi-interface</module-name>
            <access-operations>read</access-operations>
            <action>permit</action>
            <comment>Permit Read Access on "ipi-interface" Module for Group "PRIV2"</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

An `edit-config` RPC is used to **deny read access** to the **ipi-interface** module for the user group **PRIV2** by configuring deny rule in the YANG module.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>DenyReadInterfaces</name>
            <module-name>ipi-interface</module-name>
            <access-operations>read</access-operations>
            <action>deny</action>
            <comment>Deny Read Access on "ipi-interface" Module for Group "PRIV2"</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

An `edit-config` RPC is used to **permit read access** to **all modules** for the user group **PRIV2** by configuring YANG module rule.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>PermitReadAllModules</name>
            <module-name>*</module-name>
            <access-operations>read</access-operations>
            <action>permit</action>
```

```
            <comment>Permit Read Access on all Modules for Group "PRIV2"</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

An `edit-config` RPC is used to **permit all operations** on the **ipi-interface** module for the user group **PRIV2** by configuring rule in the YANG module.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>PermitAllOperationsInterfaces</name>
            <module-name>ipi-interface</module-name>
            <access-operations>*</access-operations>
            <action>permit</action>
            <comment>Permit all operations on "ipi-interface" Module for Group "PRIV2"</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

An `edit-config` RPC is used to **permit both read and exec access** to the **ipi-interface** module for the user group **PRIV2** by defining a rule in the YANG module.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>PermitReadExecInterfaces</name>
            <module-name>ipi-interface</module-name>
            <access-operations>read exec</access-operations>
            <action>permit</action>
            <comment>Permit Read and exec Access on "ipi-interface" Module for Group
"PRIV2"</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

NACM RPCs for RPC Rule

An `edit-config` RPC is used to **permit** the `get-config` operation for the user group **PRIV2** by adding an exec permission rule in the YANG module.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>permit-get-config-rpc</name>
            <rpc-name>get-config</rpc-name>
            <access-operations>exec</access-operations>
            <action>permit</action>
            <comment>Permit get-config rpc for PRIV2 group</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

An `edit-config` RPC is used to **deny** the `get-config` operation for the user group **PRIV2** by configuring a rule in the YANG module with `access-operations` set to `exec` and `action` set to `deny`.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>deny-get-config-rpc</name>
            <rpc-name>deny-config</rpc-name>
            <access-operations>exec</access-operations>
            <action>deny</action>
            <comment>Deny get-config rpc for PRIV2 group</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

An `edit-config` RPC is used to **permit all RPC operations for the user group PRIV2**by configuring a wildcard exec rule in the YANG module.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>permit-all-rpcs</name>
            <rpc-name>*</rpc-name>
            <access-operations>exec</access-operations>
            <action>permit</action>
            <comment>Permit all rpcs for PRIV2 group</comment>
```

```
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

NACM RPCs for Notification Rule

An `edit-config` RPC is used to **permit** the `interface-link-state-change-notification` notification for the user group **PRIV2** by adding a rule in the YANG module with `access-operations` set to `read` and `action` set to `permit` for notification-name `interface-link-state-change-notification`.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>permit-interface-link-state-change-notification</name>
            <notification-name>interface-link-state-change-notification</notification-name>
            <access-operations>read</access-operations>
            <action>permit</action>
            <comment>Permit notification interface-link-state-change-notification for PRIV2
group</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

An `edit-config` RPC is used to **deny** the `interface-link-state-change-notification` notification for the user group **PRIV2** by configuring a rule in the YANG module with `access-operations` set to `read` and `action` set to `deny` .

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>deny-interface-link-state-change-notification</name>
            <notification-name>interface-link-state-change-notification</notification-name>
            <access-operations>read</access-operations>
            <action>deny</action>
            <comment>Deny notification interface-link-state-change-notification for PRIV2
group</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

An `edit-config` RPC is used to **permit all notifications** for the user group **PRIV2** by adding a wildcard rule in the YANG module with `access-operations` set to `read` and `action` set to `permit`.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>permit-all-notifications</name>
            <notification-name>*</notification-name>
            <access-operations>read</access-operations>
            <action>permit</action>
            <comment>Permit all notifications for PRIV2 group</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

NACM RPCs for Data Rule

An `edit-config` RPC is used to **permit all operations** on the `/interfaces` data path for the user group **PRIV2** by configuring a rule in the YANG module.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>permit-xpath-interfaces-all-operations</name>
            <path xmlns:ipi-interface="http://www.ipinfusion.com/yang/ocnos/ipi-interface">/ipi-
interface:interfaces</path>
            <access-operations>*</access-operations>
            <action>permit</action>
            <comment>Permit all operations for xpath interfaces for PRIV2 group</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

Note: This rule is applicable for the data path `/interfaces` and all of its descendants.

An `edit-config` RPC is used to **deny edit operations** on the `/interfaces` data path for the user group **PRIV2** by configuring a rule in the YANG module with `access-operations` set to `create`, `update`, and `delete` and `action` set to `deny`.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
```

```
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
          <rule>
            <name>deny-xpath-interfaces-edit-operations</name>
            <path xmlns:ipi-interface="http://www.ipinfusion.com/yang/ocnos/ipi-interface">/ipi-
interface:interfaces</path>
            <access-operations>create update delete</access-operations>
            <action>deny</action>
            <comment>Deny edit operations for xpath interfaces for PRIV2 group</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

Note: This rule is applicable to the data path `/interfaces` and all of its descendants.

The following rules define fine-grained access control for the user group **PRIV2** using XPath expressions, specifically targeting interface-level permissions within the ipi-interface YANG module.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
        <rule>
            <name>Deny-xpath-interfaces-eth0-edit-operations</name>
            <path xmlns:ipi-interface="http://www.ipinfusion.com/yang/ocnos/ipi-interface">/ipi-
interface:interfaces/ipi-interface:interface[ipi-interface:name="eth0"]</path>
            <access-operations>create update delete</access-operations>
            <action>deny</action>
            <comment>deny edit operations for xpath interface eth0 for PRIV2 group</comment>
          </rule>
          <rule>
            <name>permit-xpath-interfaces-edit-operations</name>
            <path xmlns:ipi-interface="http://www.ipinfusion.com/yang/ocnos/ipi-interface">/ipi-
interface:interfaces</path>
            <access-operations>create update delete</access-operations>
            <action>permit</action>
            <comment>Permit edit operations for xpath interfaces for PRIV2 group</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

Purpose of Rules

This example demonstrates how to restrict edit operations (`create`, `update`, `delete`) only on interface `eth0` while allowing those operations on all other interfaces for users in NACM group PRIV2.

Rules Breakdown

- The **first rule** denies edit operations specifically on the interface node where `name = "eth0"`.

- The **second rule** permits edit operations on all interfaces under the **XPath** `/ipi-interface:interfaces`.

Rule Order

- NACM rules are evaluated in the order they appear.
- In this example, the deny rule comes first, so when the server evaluates access for eth0, it finds a match and denies the operation before reaching the permit rule.
- If the permit rule is placed before the deny rule, the server would match it first (because `/ipi-interface:interfaces includes eth0` as a descendant) and would therefore incorrectly allow edit operations on `eth0`.

Best Practice

- Always define more specific rules (e.g., for a particular interface) before more general ones (e.g., for the entire interfaces list).
- This ensures that exceptions are enforced before broader access is granted.

Rule Insertion Control

To insert a NACM rule at a specific position (e.g., before or after another rule), you can use the `yang:insert` attribute as explained in Ordered Rule Management with Yang:Insert of this guide.

Edit-config RPC to **Permit All Operations** on /interfaces/interface, Except Interfaces start with "eth", for user group **PRIV2**.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        <rule-list>
          <name>PRIV2-rules</name>
          <group>PRIV2</group>
        <rule>
            <name>Deny-xpath-interfaces-eth-edit-operations</name>
            <path xmlns:ipi-interface="http://www.ipinfusion.com/yang/ocnos/ipi-interface">/ipi-
interface:interfaces/ipi-interface:interface[ipi-interface:name='eth.*']</path>
            <access-operations>create update delete</access-operations>
            <action>deny</action>
            <comment>deny edit operations for xpath interface strat with eth for PRIV2
group</comment>
          </rule>
          <rule>
            <name>permit-xpath-interfaces-edit-operations</name>
            <path xmlns:ipi-interface="http://www.ipinfusion.com/yang/ocnos/ipi-interface">/ipi-
interface:interfaces</path>
            <access-operations>create update delete</access-operations>
            <action>permit</action>
            <comment>Permit edit operations for xpath interfaces for PRIV2 group</comment>
          </rule>
        </rule-list>
      </nacm>
    </config>
  </edit-config>
</rpc>
```

Enforce Access Control for OpenConfig Data Models

- NetConf Access Control Model (NACM) enforce access control validation for OpenConfig YANG data models when OpenConfig translation is enabled.
- This ensures that only authorized users can access, modify, or execute configuration and operational data from OpenConfig models in a network device or controller.
- Admin, ocnos, or root users can configure NACM rules for OpenConfig data models when OpenConfig translation is enabled, as explained in the previous section.

Ordered Rule Management with Yang:Insert

- NACM's `rule-list` and `rule` elements are defined as ordered-by user, meaning that administrators must have the ability to define rule precedence explicitly.
- Below are examples demonstrating how to insert rules using `yang:insert`:

Initial NACM Rule Configuration:

```
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
    <rule-list>
      <name>admin-rules</name>
      <rule>
        <name>existing-rule</name>
        <action>permit</action>
        <comment>Existing rule in NACM</comment>
      </rule>
    </rule-list>
  </nacm>
```

Adding a New Rule After an Existing Rule:

```
<edit-config>
  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
      <rule-list>
        <name>admin-rules</name>
        <rule
             xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"
             xmlns:yang="urn:ietf:params:xml:ns:yang:1"
             yang:insert="after"
             yang:key="[name='existing-rule']">
          <name>new-rule</name>
          <action>deny</action>
          <comment>New rule inserted after existing-rule</comment>
        </rule>
      </rule-list>
    </nacm>
  </config>
</edit-config>
```

Adding a Rule Before an Existing Rule:

```
<edit-config>
  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
      <rule-list>
        <name>admin-rules</name>
        <rule
             xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"
             xmlns:yang="urn:ietf:params:xml:ns:yang:1"
             yang:insert="before"
```

© 2026 IP Infusion Inc. Proprietary

```
            yang:key="[name='existing-rule']>
          <name>high-priority-rule</name>
          <action>deny</action>
          <comment>Inserted before existing-rule</comment>
        </rule>
      </rule-list>
    </nacm>
  </config>
</edit-config>
```

Adding a Rule at first:

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>admin-rules</name>
     <rule xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"
      xmlns:yang="urn:ietf:params:xml:ns:yang:1"
      yang:insert="first">
      <name>allow-get-config</name>
      <rpc-name>get-config</rpc-name>
      <access-operations>*</access-operations>
      <action>permit</action>
      <comment>Allow get-config rpc for PRIV1 group</comment>
    </rule>
 </rule-list>
</nacm>
```

Adding a Rule at last:

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>admin-rules</name>
     <rule xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"
      xmlns:yang="urn:ietf:params:xml:ns:yang:1"
      yang:insert="last">
      <name>allow-get-config</name>
      <rpc-name>get-config</rpc-name>
      <access-operations>*</access-operations>
      <action>permit</action>
      <comment>Allow get-config rpc for PRIV1 group</comment>
    </rule>
 </rule-list>
</nacm>
```

Using NETCONF to Manage NACM Rules

- Apply rules using the `<edit-config>` RPC.
- Validate rule application using `<get-config>`.
- Verify rule enforcement by performing specific operations.
- Modify or delete rules using `<edit-config>` with the appropriate XPath.

Troubleshooting

- Ensure the rule list name and paths are correct.
- Confirm users are part of the correct groups. Check netconfd logs for errors.
- Enable NETCONF debug logs as given below and check

```
yangcli ocnos@0> set-log-level log-level=debug4
```

Note:

- To check if the rule is applied or not - Use `<get-config>` to retrieve the NACM configuration.
- To delete a rule - Use `<edit-config>` with the operation delete on the target rule path.

Limitations

- NACM relies on NetConf transport layer for user authentication.
- User-to-group mapping is dependent on admin.
- Performance impacts on large configurations, NACM rule evaluation can add processing overhead, especially for fine-grained data node checks.

Glossary

The following provides definitions for key terms used throughout this document.

Access Control	A security feature provided by the server that allows an administrator to restrict access to a subset of all protocol operations and data, based on various criteria.
NETCONF Access Control Model (NACM)	A model used to configure and monitor the access control procedures desired by the administrator to enforce a particular access control policy.
YANG Module	YANG (Yet Another Next Generation) is a data modeling language standardized by the IETF (RFC 7950). It is a structured, machine-readable file that defines the data model used by network management protocols notably NETCONF.
Remote Procedure Calls (RPC) rule	In the context of NACM (Network Configuration Access Control Model) it is an access control rule that determines whether a user is allowed or denied permission to run specific RPCs or actions defined in YANG modules.
NETCONF	Network Configuration Protocol
RPC	Remote Procedure Call
TLS	Transport Layer Security
SSH	Secure Shell
PRIV1	Highest Privilege user group (admin and ocnos users)

© 2026 IP Infusion Inc. Proprietary

IMAGE UPGRADE BY TRAFFIC DIVERSION (IUTD)

Overview

Image Upgrade by Traffic Diversion (IUTD) is an uncommon method used for traffic diversion, during the upgrade or installation process to ensure continuous network operation by temporary re-routing network traffic to the different node and restoring the traffic flow to the updated node after verifying the service.

Feature Characteristics

- Minimal Traffic Loss: The IUTD process enables node software updates, resulting in minimum traffic loss during the brief convergence time required while switching traffic paths.
- Traffic Handling: Traffic is manually diverted/rerouted (using `edit-config`) to the redundant mode before the upgrade and restored only after verifications are complete.
- Redundancy Dependent: Requires the presence of a redundant node/path to handle the full traffic load during the maintenance window.
- Service Monitoring: A NetConf filter checks the traffic status of specific services, such as VxLAN.
- Real-Time Alerts: NetConf sends notifications whenever a service starts or stops carrying traffic, to enable quick operational response.
- Controller Visibility: The controller receives both NetConf and SNMP notifications upon successful image upgrade completion, providing full visibility into the status.
- Error Intimation: Failure occurs, error will be reported. OS download cancels and no impact on the system (System continues to be in the original condition).

Prerequisites

Implementing the IUTD feature requires a NetConf client, such as Netopeer2, capable of utilizing the call-home functionality.

This feature applies exclusively to the VxLAN Multihoming (MH) scenario within a DC-CLOS architecture.

© 2026 IP Infusion Inc. Proprietary

Sequence of IUTD process

Initial Scenario & Traffic Diversion (Service UP -> DOWN)

This phase establishes the secure control channel and safely moves traffic away from the node to be upgraded.

- **Initiation & Session:** An API or GUI initiates the process, leading the Controller to establish a NetConf session with the Leaf node.
- **Notification Setup:** The Controller subscribes to receive notifications and sends configuration to enable the nodes NetConf Callhome feature.
  - This requirement implements a function designed to notify the controller of changes in Service Status. This notification has two states:
    - **Service Status Down**: Verifies successful traffic diversion by checking traffic using counters, interface status, and traffic rates to assure traffic is DOWN.
    - **Service Status Up**: Verifies successful traffic restoration by checking traffic using counters, interface status, and traffic rates to assure service flows are fully established.
- **Traffic Monitoring Start:** The Controller invokes the `start-service-tracking` RPC to instruct the node (OcNOS) to monitor the status of a specific service. This tracking is initially set to detect the service going DOWN.
- **Traffic Diversion:** The Controller sends the necessary configuration changes to the node to bypass (divert) traffic.
- **Diversion Confirmation:** The node's NSM (Network Services Module) checks traffic counters and routes. Once traffic is confirmed to be fully diverted, it sends a `service-tracking-status` notification indicating Service-DOWN.

Installation and OS Update

After the node safely isolated, the actual OS update is performed, followed by a controlled reboot:

- **Image Download:** The Controller invokes the `sys-update-get` RPC for the node to download the new OS image. The node notifies the Controller when the download finishes (`sys-update-download-status`).
- **Image Installation:** The Controller invokes the `sys-update-install` RPC.
- **Pre-Reboot Alert:** The node sends a notification (`sys-update-installation-status`) warning that the "System is going down for upgrade" just before the reboot.
- **Session Close & Callhome Setup:** The old NetConf session closes. The Controller switches to listen mode to wait for the node's NetConf Callhome to reconnect after the reboot.

Verification and Traffic Restoration

The node reconnects with the new OS, and the Controller confirms stability before restoring traffic.

- **Re-establishment & Re-subscription:** The node initiates the NetConf Callhome connection, establishing a new session. The Controller immediately resubscribes to ensure no cached notifications are missed.

© 2026 IP Infusion Inc. Proprietary

- **Upgrade Success Confirmation:** The node sends both the cold start SNMP trap notification and an SNMP trap with a "Success/Failure" message.
- **Restore Monitoring Start:** The Controller invokes the `start-service-tracking` RPC again for the same service (VxLAN VNI 5), this time to detect the service coming UP.
  - Refer to the "VxLAN EVPN Multi-Homing Configuration'' for feature to ensure redundant traffic paths during the upgrade.
- **Traffic Restoration:** The Controller edits the configuration to restore traffic to the upgraded node.
- **Restoration Confirmation:** The node's NSM checks for established traffic flows. Once the service is stable and traffic is confirmed flowing, it sends a final `service-tracking-status` notification indicating Service-UP.
- **Save Configuration:** The Controller saves the final, running configuration to the start-up configuration on the node.

© 2026 IP Infusion Inc. Proprietary

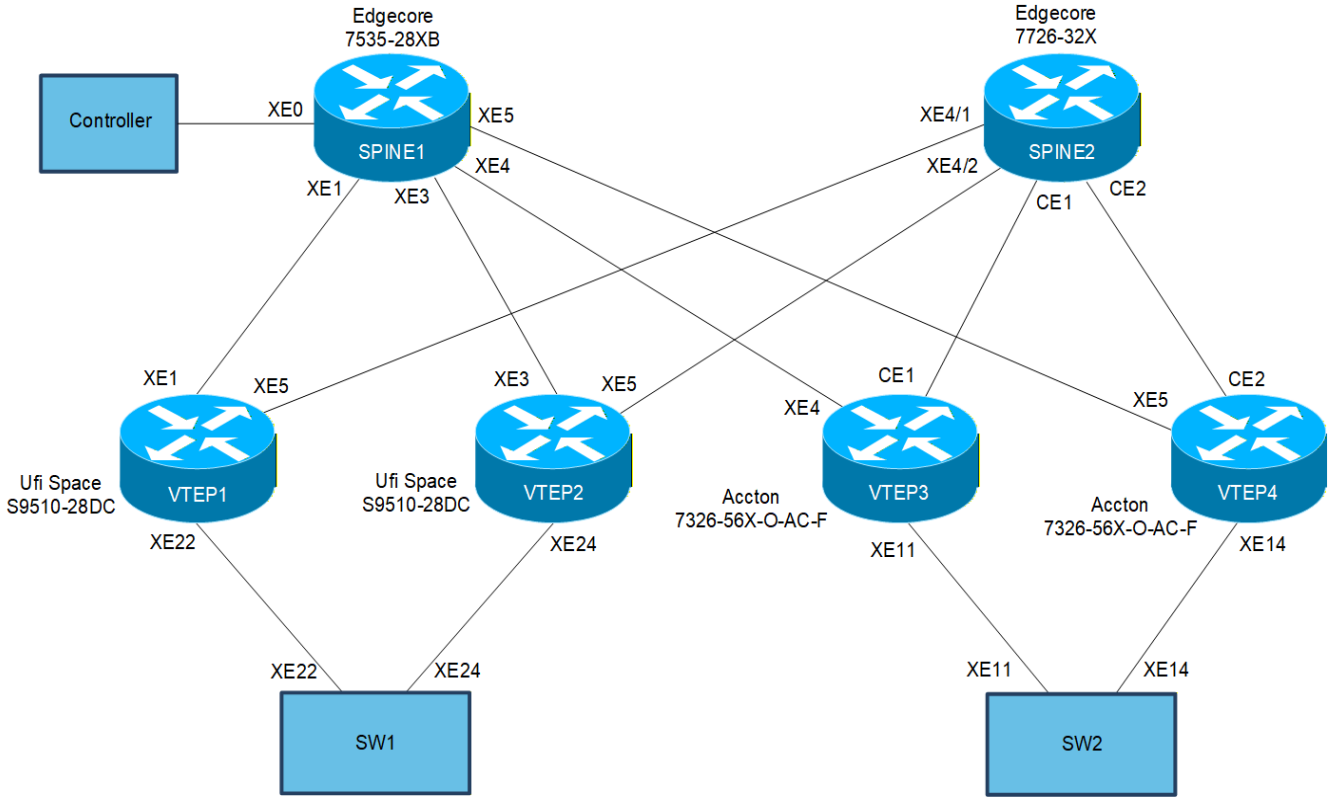
Judgement Criteria

To confirm that the actions done to divert/re-route traffic has taken effect based on two process:

- In-Band Management Scenario - Since we cannot shut down network-side interfaces, and routes cannot be removed. we must use traffic counters to confirm successful traffic diversion. This method requires a key assumption: steady Unicast traffic must be flowing through the nodes. Through this we judge the effectiveness of the traffic steering actions by observing changes in the traffic rate. In the case of a Leaf node, we will also use the access interface state as a primary indicator.
- Out-Band Management Scenario - the judgment is common for leaf and spine. All the interfaces except SVI, loopback and management interfaces will be shutdown before updating the OS and it will be necessary to check the interface status and the traffic.

Topology

Figure 4. Traffic diversion management scenario for OS installation and update



In-Band Management Scenario

Judgement logic for detecting changes in service status - Leaf

This scenario assumes a network with only multi-homed leaves and uses the access interface state to control traffic flow, ensuring the network-facing interfaces remain up to maintain the in-band management connection.

Traffic Diversion (Service UP to DOWN)

The goal is to move all traffic off the Leaf node by shutting down the customer-facing link.

- Preparation & Tracking:
  - The Controller invokes the `start-service-tracking` RPC, instructing the node to monitor for the `Service-DOWN` status with a specified `service-timeout` (e.g., 60 seconds).
  - Pre-Check: The node first records all access interfaces currently carrying non-zero traffic; if no traffic is flowing, the tracking command returns an error.
- Traffic Reroute: The Controller sends an `edit-config` operation to shut down the access-interface of the VXLAN instance.
- Optional Optimizations (Minimize Loss):
  - Enable BFD between the Leaf and the Compute Node so the compute node can quickly switch traffic to the multihome peer upon link shutdown.
  - Set the BGP advertisement interval to 0 to ensure the BGP withdraw message reaches remote leaves faster, preventing incoming traffic loss.
- Isolation Check (Service DOWN Success): The node continuously verifies that:
  - LACP and the VXLAN interface are DOWN.
  - Traffic is not flowing **(Tx/Rx counts = 0)** for a predefined time (e.g., 10 seconds).
- Outcome: If all conditions are met before the timeout, a `SERVICE_STATUS` notification is sent with `Success`. If the timeout is reached, it sends Failure.

Traffic Diversion (Service DOWN to UP)

The goal is to bring the updated node back into service and verify traffic flow.

- Tracking Start: The Controller invokes the `start-service-tracking` RPC to monitor for the Service-UP status.
- Traffic Restore: The Controller sends an `edit-config` operation to perform a `no shutdown` on the VXLAN access interfaces.
- Restoration Check (Service UP Success): The node continuously verifies that:
  - LACP and the VXLAN interface are UP.
  - Traffic is flowing **(Tx/Rx counts > 0)** on all access interfaces that previously carried non-zero traffic.
- Outcome: If all conditions are met before the timeout, a `SERVICE_STATUS` notification is sent with `Success`. If the timeout is reached without confirmed traffic flow, it sends Failure.

Judgement logic for detecting changes in service status - Spine

Traffic diversion on the Spine node is achieved by increasing the routing cost of protocols; restoration involves resetting the cost to default. Network interfaces remain active to preserve the in-band management connection.

Key Traffic Definitions

- **Pre-diversion-incoming-traffic-rate**: The normal, steady traffic rate measured just before diversion starts.
- **Minimum-traffic-threshold-rate (100 pps)**: A predefined low rate used to confirm user traffic is gone.
  - Traffic counters never reach zero because essential control plane traffic (BGP,ISIS,OSPF, etc.) continues to flow.
  - If traffic drops below this threshold, it confirms successful diversion, leaving only control plane traffic.

Traffic Diversion (Service UP to DOWN)

The goal is to move user traffic off the Spine node by increasing routing cost and confirming the traffic rate drops below a minimum threshold within a 20-second timeout.

- Preparation & Tracking:
  - The Controller invokes the `start-service-tracking` RPC for Service-DOWN status, monitoring the EBGP service with a 20-second timeout.
  - The Spine node records the current traffic rate as `pre-diversion-incoming-traffic-rate` (retained even after reboot).
  - Success Pre-Condition: The recorded traffic rate must be greater than 10 times the `minimum-traffic-threshold-rate` for the RPC to proceed; otherwise, it returns a failure.
- Traffic Diversion: The Controller uses `edit-config` to divert traffic by increasing the routing cost of protocols like OSPF, ISIS, or BGP. Refer section: "Routing cost change in protocols" for more details.
- Isolation Check: The system verifies that the traffic rate falls from the `pre-reroute-incoming-traffic-rate` to below the `minimum-traffic-threshold-rate`.
- Outcome: If the traffic drops below the threshold before the timeout, the node sends a `SERVICE_STATUS` notification indicating `Service-DOWN` with Success.

Traffic Diversion (Service DOWN to UP)

The goal is to restore the original routing cost and confirm that user traffic returns to normal levels within 20 seconds.

- Preparation & Tracking:
  - The Controller invokes the `start-service-tracking` RPC for Service-UP status, again with a 20-second timeout.
- Traffic Restore: The Controller uses `edit-config` to restore traffic by changing the routing cost back to the original (default) values.
- Restoration Check: The system verifies that the traffic rate rises above 50% (1/2) of the stored `pre-diversion-incoming-traffic-rate` and sustains this rate for at least 10 seconds.
- Outcome: If the traffic rate recovery is confirmed before the timeout, the node sends a `SERVICE_STATUS` notification indicating `Service-UP` with Success.

IUTD Spine Logic: Assumptions and Limitations

One of the crucial block of assumptions and limitations for the Spine Node's IUTD logic, particularly when relying on traffic counters.

- Assumptions for Tracking Initiation
  - Sufficient Baseline Traffic: Service tracking can only be initiated if the existing traffic load meets a minimum threshold: the pre-diversion-incoming-traffic-rate must be greater than 10 times the minimum-traffic-threshold-rate.
- Limitations and Potential Failures
  - Unstable Traffic Pattern (False Result): If the network's external traffic pattern drastically changes and the traffic rate drops to the minimum-traffic-threshold-rate before the routing metrics are intentionally changed, the system will generate a false success result.
  - Traffic Failure to Restore (False Failure): During the Service-DOWN to UP tracking, if the traffic rate fails to restore to at least half (1/2) of the original pre-diversion-incoming-traffic-rate due to external traffic pattern changes (not a problem with the node), the service-up tracking will incorrectly report a Failure.

© 2026 IP Infusion Inc. Proprietary

    - SVI/Trunk Interface Limitation: Service tracking may not function correctly if the underlying routing protocols rely on Switched Virtual Interfaces (SVIs) or trunk interfaces. This is because SVIs under routing protocols may not properly increment the necessary traffic counters used to calculate the traffic rate, leading to inaccurate measurements.
- Recommendation
    - Interface Usage: It is recommended to use physical interfaces as link interfaces between network equipment to ensure accurate traffic counting and reliable service tracking.

Routing cost change in protocols

Following are the details of how to change routing costs in underlay protocol modules in order to divert traffic. To restore the reverse operation needs to be done.

Note: All operations initiated from the controller will be via Netconf. However the CLI examples are given for reference and ease of understanding.

OSPF

```
router ospf
 max-metric router-lsa
!
```

Netconf payload:

```
<ospfv2 xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-ospf">
   <processes>
   <process>
      <ospf-id>0</ospf-id>
      <config>
         <ospf-id>0</ospf-id>
      </config>
   <max-metric>
   <config>
         </enable-max-router-lsa>
   </config>
   </max-metric>
   </process>
   </processes>
   </ospfv2>
```

ISIS

```
router isis 1
 set-overload-bit
!
```

```
<isis xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-isis">
  <isis-instances>
    <isis-instance>
      <instance>1</instance>
      <lsp-over-load>
        <config>
          <enabled></enabled>
        </config>
      </lsp-over-load>
    </isis-instance>
  </isis-instances>
</isis>
```

BGP

The BGP considered in only the eBGP. To re-route BGP, the attribute that will be used is MED (Multi-Exit Discriminator). It is used in best path selection by BGP. MED is compared after BGP attributes weight, local preference, AS-path and origin have been compared and are equal. MED comparison is done only among paths from the same autonomous system (AS). Use BGP always-compare-med command to allow comparison of MEDs from different ASs.

This config can be already there:

```
router bgp 1
 bgp always-compare-med
```

Then, for diverting the traffic the following operation can be done:

```
route-map rmap1 permit
 set metric 300
router bgp 1
 address-family ipv4 unicast
  neighbor 6.6.6.6 route-map rmap1 out
```

Netconf payload:

```
<routemaps xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-routemap">
  <routemap>
    <routemap-name>rmap1</routemap-name>
    <sequence-id>10</sequence-id>
    <config>
      <routemap-name>rmap1</routemap-name>
      <sequence-id>10</sequence-id>
      <action>permit</action>
    </config>
    <set-action>
      <config>
        <metric-value>300</metric-value>
      </config>
    </set-action>
  </routemap>
</routemaps>
<bgp xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-bgp">
  <bgp-instance>
    <bgp-as>1</bgp-as>
    <peer>
      <peer-address>6.6.6.6</peer-address>
      <address-family>
        <afi>ipv4</afi>
        <safi>unicast</safi>
        <route-map-filter>
          <route-map-direction>out</route-map-direction>
          <config>
            <route-map-direction>out</route-map-direction>
            <route-map-name>rmap1</route-map-name>
          </config>
        </route-map-filter>
      </address-family>
    </peer>
  </bgp-instance>
</bgp>
```

Out-Band Management Scenario

This scenario uses the Interface (IF) status as the service to track, leveraging administrative shutdowns to achieve traffic isolation for the upgrade. This is valid for both Leaf and Spine.

An assumption in this logic is there are no topology changes happening during upgrades, as the LINK_UP_IF_LIST will not match.

Traffic Diversion (Service UP to DOWN)

The goal is to administratively shut down all traffic-carrying interfaces and confirm their link status goes down within the 20-second timeout.

- Preparation & Tracking:
  - The Controller invokes the `start-service-tracking` RPC for Service-DOWN status, monitoring the IF service with a 20-second timeout.
  - The system persistently saves a list of all currently UP interfaces as `LINK_UP_IF_LIST` for later restoration checks.
- Traffic Diversion: The Controller uses `edit-config` operation to shut down all interfaces, excluding essential management interfaces (management, SVI, and loopback).
- Optional Optimizations (Minimize Loss):
  - Enable BFD on compute node links for fast traffic switchover upon link shutdown.
  - Set BGP advertisement interval to 0 to ensure fast route withdrawal at remote leaves, minimizing incoming traffic loss.
- Isolation Check (Service DOWN Success): The node continuously verifies that:
  - All interfaces have an admin status of DOWN
  - All interfaces saved in `LINK_UP_IF_LIST` have a link status of DOWN.
- Outcome: If all conditions are met before the timeout, a `SERVICE_STATUS` notification is sent with Success. If the timeout is reached, it sends Failure.

Traffic Diversion (Service DOWN to UP)

The goal is to bring the interfaces back up and confirm their link status returns to operational.

- Tracking Start: The Controller invokes the `start-service-tracking` RPC for Service-UP status.
- Traffic Restore: The Controller sends an `edit-config` operation to perform a `no shutdown` on all interfaces.
- Restoration Check (Service UP Success): The node continuously verifies that:
  - All interfaces are UP again.
  - Specifically, all interfaces listed in the `LINK_UP_IF_LIST` are confirmed to be UP again.
- Outcome: If all conditions are met before the timeout, a `SERVICE_STATUS` notification is sent with Success. If the timeout is reached, it sends Failure.

Datamodel for Service Tracking

Datamodel name: ipi-service-tracking

NetConf RPC

- start-service-tracking (New)
  - Service-type : VxLAN/EBGP/OSPF/ISIS/IF
  - Operational-status-check : Up/Down
  - Service-Timeout: <range of 20-600 seconds>

Notification

- service-tracking-status: (New)
  - Type state

- Severity: Info
- Attributes:
  - Service-Type: VxLAN/EBGP/OSPF/ISIS/IF
  - Operational-status-check : Up/Down
  - Status: Success/Fail

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2023-09-05T10:14:47Z</eventTime>
  <severity>info</severity>
  <eventClass>state</eventClass>
  <service-tracking-status xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-service-tracking">
    <service-Type>VXLAN</service-Type>
    <operational-status-check>UP</<operational-status-check>>
    <status>Success</status>
  </service-tracking-status>
</notification>
```

Netconf get

There is a list that keeps the last status of the a service tracking. The entries are filled for each service type and starts to appear in get after the start-service-tracking RPC is issued for the first time, then keeps always the last status.

```
rpc-reply {
  data {
    services-tracking {
      service-tracking VXLAN {
        service-type VXLAN
        state {
          service-type VXLAN
          operational-status-check DOWN
          status FAILURE
        }
      }
      service-tracking ISIS {
        service-type ISIS
        state {
          service-type ISIS
          operational-status-check UP
          status IN-PROGRESS
        }
      }
    }
  }
}
```

Validation

To start the service tracking up -> down, IF all Leaf

```
user-rpc --content /root/rpcs/rpc-svc-track-if-down.xml --rpc-timeout 900

root@lab10:/home/user/IUTD/rpcs# cat rpc-svc-track-if-down.xml
<start-service-tracking xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-service-tracking">
  <service-type>IF</service-type>
  <operational-status-check>DOWN</operational-status-check>
  <service-timeout>120</service-timeout>
</start-service-tracking>
root@lab10:/home/user/IUTD/rpcs#
```

To edit configuration to shut down all the interface, up -> down, IF all leaf

```
edit-config --target candidate --config=/root/LOGs_IUTD/Leaf_OutBand/config-shut-if-all-
leaf1.xml --rpc-timeout 900


root@lab10:/home/user/IUTD/LOGs_IUTD/Leaf_OutBand# cat config-shut-if-all-leaf1.xml
<interfaces xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-interface">
  <interface>
    <name>cd0</name>
    <config>
      <name>cd0</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>cd1</name>
    <config>
      <name>cd1</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>ce2</name>
    <config>
      <name>ce2</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>ce3</name>
    <config>
      <name>ce3</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe4</name>
    <config>
      <name>xe4</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe5</name>
    <config>
      <name>xe5</name>
      <shutdown></shutdown>
    </config>
```

© 2026 IP Infusion Inc. Proprietary

```
</interface>
<interface>
  <name>xe6</name>
  <config>
    <name>xe6</name>
    <shutdown></shutdown>
  </config>
</interface>
<interface>
  <name>xe7</name>
  <config>
    <name>xe7</name>
    <shutdown></shutdown>
  </config>
</interface>
<interface>
  <name>xe8</name>
  <config>
    <name>xe8</name>
    <shutdown></shutdown>
  </config>
</interface>
<interface>
  <name>xe9</name>
  <config>
    <name>xe9</name>
    <shutdown></shutdown>
  </config>
</interface>
<interface>
  <name>xe10</name>
  <config>
    <name>xe10</name>
    <shutdown></shutdown>
  </config>
</interface>
<interface>
  <name>xe11</name>
  <config>
    <name>xe11</name>
    <shutdown></shutdown>
  </config>
</interface>
<interface>
  <name>xe12</name>
  <config>
    <name>xe12</name>
    <shutdown></shutdown>
  </config>
</interface>
<interface>
  <name>xe13</name>
  <config>
    <name>xe13</name>
    <shutdown></shutdown>
  </config>
</interface>
<interface>
  <name>xe14</name>
  <config>
    <name>xe14</name>
    <shutdown></shutdown>
  </config>
</interface>
<interface>
  <name>xe15</name>
  <config>
    <name>xe15</name>
```

```
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe16</name>
    <config>
      <name>xe16</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe17</name>
    <config>
      <name>xe17</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe18</name>
    <config>
      <name>xe18</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe19</name>
    <config>
      <name>xe19</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe20</name>
    <config>
      <name>xe20</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe21</name>
    <config>
      <name>xe21</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe22</name>
    <config>
      <name>xe22</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe23</name>
    <config>
      <name>xe23</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe24</name>
    <config>
      <name>xe24</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe25</name>
```

```
    <config>
      <name>xe25</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe26</name>
    <config>
      <name>xe26</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe27</name>
    <config>
      <name>xe27</name>
      <shutdown></shutdown>
    </config>
  </interface>
  <interface>
    <name>po1</name>
    <config>
      <name>po1</name>
      <shutdown></shutdown>
    </config>
  </interface>
</interfaces>
```

To commit the configuration

```
commit --rpc-timeout 900
```

copy running-configuration to startup-configuration

```
copy-config --source running --target startup --rpc-timeout 900
```

To verify interface counter rate and interface status and other show commands using CMLSH

```
SP-LEAF1#sh interface counter rate mbps
+------------------+--------------+-------------+--------------+-------------+
|     Interface    |    Rx mbps   |    Rx pps   |    Tx mbps   |    Tx pps   |
+------------------+--------------+-------------+--------------+-------------+

SP-LEAF1#sh run
!
! Software version: UFI_S9510-28DC-OcNOS-SP-PLUS-7.0.0.137-Alpha 10/13/2025 17:3
9:59
!
! Last configuration change at 15:03:58 UTC Wed Oct 15 2025 by root
!
!
feature netconf callhome
 callhome enable vrf management
  reconnect enable
  retry-max-attempts 10
  retry-interval 100
  callhome server netopeer 10.16.99.114
  exit-netconf-callhome-vrf
 !
!
service password-encryption
!
logging console disable
```

```
logging monitor disable
logging logfile device_debug_log 5
logging level all 5
!
!
snmp-server enable traps link linkDown
snmp-server enable traps link linkUp
!
load-balance enable
hardware-profile filter egress-ipv4 enable
hardware-profile filter vxlan enable
hardware-profile filter vxlan-mh enable
hardware-profile statistics voq-full-color enable
hardware-profile statistics cfm-ccm enable
!
bfd interval 10 minrx 10 multiplier 3
!
qos enable
!
hostname SP-LEAF1
ip domain-lookup vrf management
ip name-server vrf management 10.16.10.23
tfo Disable
errdisable cause stp-bpdu-guard
feature dns relay
ip dns relay
ipv6 dns relay
!
nvo vxlan enable
!
evpn esi hold-time 100
!
evpn vxlan multihoming enable
!
ip vrf management
!
mac vrf vrf1
 rd 1.1.1.1:1
 route-target both 100:1
!
mac vrf vrf10
 rd 1.1.1.1:10
 route-target both 100:10
!
mac vrf vrf2
 rd 1.1.1.1:2
 route-target both 100:2
!
mac vrf vrf3
 rd 1.1.1.1:3
 route-target both 100:3
!
mac vrf vrf4
 rd 1.1.1.1:4
 route-target both 100:4
!
mac vrf vrf5
 rd 1.1.1.1:5
 route-target both 100:5
!
mac vrf vrf6
 rd 1.1.1.1:6
 route-target both 100:6
!
mac vrf vrf7
 rd 1.1.1.1:7
 route-target both 100:7
!
```

```
mac vrf vrf8
 rd 1.1.1.1:8
 route-target both 100:8
!
mac vrf vrf9
 rd 1.1.1.1:9
 route-target both 100:9
!
nvo vxlan vtep-ip-global 1.1.1.1
!
nvo vxlan id 100 ingress-replication
 vxlan host-reachability-protocol evpn-bgp vrf1
!
nvo vxlan id 101 ingress-replication
 vxlan host-reachability-protocol evpn-bgp vrf2
!
nvo vxlan id 102 ingress-replication
 vxlan host-reachability-protocol evpn-bgp vrf3
!
nvo vxlan id 103 ingress-replication
 vxlan host-reachability-protocol evpn-bgp vrf4
!
nvo vxlan id 104 ingress-replication
 vxlan host-reachability-protocol evpn-bgp vrf5
!
nvo vxlan id 105 ingress-replication
 vxlan host-reachability-protocol evpn-bgp vrf6
!
nvo vxlan id 106 ingress-replication
 vxlan host-reachability-protocol evpn-bgp vrf7
!
nvo vxlan id 107 ingress-replication
 vxlan host-reachability-protocol evpn-bgp vrf8
!
nvo vxlan id 108 ingress-replication
 vxlan host-reachability-protocol evpn-bgp vrf9
!
nvo vxlan id 109 ingress-replication
 vxlan host-reachability-protocol evpn-bgp vrf10
!
interface po1
 shutdown
!
interface po100
 switchport
 load-interval 30
 evpn multi-homed system-mac aaaa.aaaa.aaa1
!
interface po100.100 switchport
 encapsulation dot1q 100
 rewrite pop
 access-if-evpn
  map vpn-id 100
!
interface po100.101 switchport
 encapsulation dot1q 101
 rewrite pop
 access-if-evpn
  map vpn-id 101
!
interface po100.102 switchport
 encapsulation dot1q 102
 rewrite pop
 access-if-evpn
  map vpn-id 102
!
interface po100.103 switchport
 encapsulation dot1q 103
```

```
 rewrite pop
 access-if-evpn
  map vpn-id 103
!
interface po100.104 switchport
 encapsulation dot1q 104
 rewrite pop
 access-if-evpn
  map vpn-id 104
!
interface po100.105 switchport
 encapsulation dot1q 105
 rewrite pop
 access-if-evpn
  map vpn-id 105
!
interface po100.106 switchport
 encapsulation dot1q 106
 rewrite pop
 access-if-evpn
  map vpn-id 106
!
interface po100.107 switchport
 encapsulation dot1q 107
 rewrite pop
 access-if-evpn
  map vpn-id 107
!
interface po100.108 switchport
 encapsulation dot1q 108
 rewrite pop
 access-if-evpn
  map vpn-id 108
!
interface po100.109 switchport
 encapsulation dot1q 109
 rewrite pop
 access-if-evpn
  map vpn-id 109
!
interface cd0
 shutdown
!
interface cd1
 shutdown
!
interface ce2
 load-interval 30
 ip address 90.90.90.1/24
 shutdown
!
interface ce3
 shutdown
!
interface eth0
 ip vrf forwarding management
 ip address dhcp
!
interface lo
 ip address 127.0.0.1/8
 ip address 1.1.1.1/32 secondary
 ipv6 address ::1/128
!
interface lo.management
 ip vrf forwarding management
 ip address 127.0.0.1/8
 ipv6 address ::1/128
!
```

```
interface xe4
 shutdown
!
interface xe5
 speed 10g
 load-interval 30
 ip address 60.60.60.1/24
 shutdown
!
interface xe6
 shutdown
!
interface xe7
 shutdown
!
interface xe8
 shutdown
!
interface xe9
 shutdown
!
interface xe10
 shutdown
!
interface xe11
 shutdown
!
interface xe12
 shutdown
!
interface xe13
 shutdown
!
interface xe14
 shutdown
!
interface xe15
 shutdown
!
interface xe16
 shutdown
!
interface xe17
 shutdown
!
interface xe18
 shutdown
!
interface xe19
 shutdown
!
interface xe20
 shutdown
!
interface xe21
 shutdown
!
interface xe22
 speed 10g
 channel-group 100 mode active
 shutdown
!
interface xe23
 shutdown
!
interface xe24
 shutdown
!
```

```
interface xe25
 shutdown
!
interface xe26
 shutdown
!
interface xe27
 shutdown
!
 exit
!
router bgp 100
 bgp router-id 1.1.1.1
 no bgp inbound-route-filter
 neighbor Overlay peer-group
 neighbor Overlay remote-as 200
 neighbor Overlay ebgp-multihop
 neighbor Overlay update-source lo
 neighbor Overlay advertisement-interval 0
 neighbor Overlay fall-over bfd multihop
 neighbor Underlay peer-group
 neighbor Underlay remote-as 200
 neighbor Underlay advertisement-interval 0
 neighbor Underlay fall-over bfd
 neighbor 5.5.5.5 peer-group Overlay
 neighbor 6.6.6.6 peer-group Overlay
 neighbor 60.60.60.2 peer-group Underlay
 neighbor 90.90.90.2 peer-group Underlay
 !
 address-family ipv4 unicast
 network 1.1.1.1/32
 max-paths ebgp 3
 neighbor Underlay activate
 neighbor Underlay allowas-in 1
 exit-address-family
 !
 address-family l2vpn evpn
 neighbor Overlay activate
 neighbor Overlay allowas-in 1
 exit-address-family
 !
 exit
!
line console 0
 exec-timeout 0
line vty 0 16
 exec-timeout 0 0
!
!
End
```

To verify service tracking is in progress

```
get --filter-subtree=/root/user/filters/filter-service-tracking.xml

root@lab10:/home/user/IUTD/filters# cat filter-service-tracking.xml
<services-tracking xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-service-tracking"/>

OUTPUT:
> get --filter-subtree=/root/nantha/filters/filter-service-tracking.xml
DATA
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <services-tracking xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-service-tracking">
    <service-tracking>
      <service-type>IF</service-type>
      <state>
        <service-type>IF</service-type>
```

```
        <operational-status-check>DOWN</operational-status-check>
        <status>SUCCESS</status>
      </state>
    </service-tracking>
```

Configuration to get the image

```
user-rpc --content /root/user/rpc-sys-update-get-sp.xml --rpc-timeout 900

root@lab10:/home/user/IUTD/rpcs# cat rpc-sys-update-get.xml
<sys-update-get xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-sys-update">
  <source-interface>eth0</source-interface>
  <url>http://1.1.1.2/OcNOS-SP-PLUS-Q2-7.0.0-172-Alpha-installer</url>
</sys-update-get>
```

To install the image

```
user-rpc --content /root/OcNOS/IUTD/rpcs/rpc-sys-update-install-dc.xml --rpc-timeout 900

root@lab10:/home/user/IUTD/rpcs# cat rpc-sys-update-install-dc.xml
<sys-update-install xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-sys-update">
  <installer-name>OcNOS-SP-PLUS-Q2-7.0.0-172-Alpha-installer</installer-name>
  <ignore-feature-check>true</ignore-feature-check>
</sys-update-install>
```

Configuration to reload/image change process started netconf connection will close and initiate the callhome in netopeer

```
#on client to reconnect with OcNOS
   exit
   ./netopeer2-cli
   listen --host 0.0.0.0 --timeout 250 --login ocnos

OUTPUT:

> listen --host 0.0.0.0 --timeout 250 --login ocnos
Waiting 250s for an SSH Call Home connection on port 4334...


The authenticity of the host '10.16.109.120' cannot be established.
ssh-ed25519 key fingerprint is 19:7a:00:cb:c1:4e:da:27:3e:c0:c2:dc:ed:32:ab:c2:02:80:a1:32.
Are you sure you want to continue connecting (yes/no)? yes
ocnos@10.16.109.120 password:
```

To start the service tracking down -> if all leaf

```
user-rpc --content /root/user/rpcs/rpc-svc-track-if-up.xml --rpc-timeout 900

cat rpc-svc-track-if-up.xml
<start-service-tracking xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-service-tracking">
  <service-type>IF</service-type>
  <operational-status-check>UP</operational-status-check>
  <service-timeout>60</service-timeout>
</start-service-tracking>
```

To edit config to no shut down all the interface, down -> up, IF all leaf

```
edit-config --target candidate --config=/root/user/LOGs_IUTD/Leaf_OutBand/noshut.xml --rpc-
timeout 900

cat noshut.xml
```

```
<interfaces xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-interface">
  <interface>
    <name>cd0</name>
    <config>
      <name>cd0</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>cd1</name>
    <config>
      <name>cd1</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>ce2</name>
    <config>
      <name>ce2</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>ce3</name>
    <config>
      <name>ce3</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe4</name>
    <config>
      <name>xe4</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe5</name>
    <config>
      <name>xe5</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe6</name>
    <config>
      <name>xe6</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe7</name>
    <config>
      <name>xe7</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe8</name>
    <config>
      <name>xe8</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe9</name>
    <config>
      <name>xe9</name>
```

```
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe10</name>
    <config>
      <name>xe10</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe11</name>
    <config>
      <name>xe11</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe12</name>
    <config>
      <name>xe12</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe13</name>
    <config>
      <name>xe13</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe14</name>
    <config>
      <name>xe14</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe15</name>
    <config>
      <name>xe15</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe16</name>
    <config>
      <name>xe16</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe17</name>
    <config>
      <name>xe17</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe18</name>
    <config>
      <name>xe18</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe19</name>
```

```
    <config>
      <name>xe19</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe20</name>
    <config>
      <name>xe20</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe21</name>
    <config>
      <name>xe21</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe22</name>
    <config>
      <name>xe22</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe23</name>
    <config>
      <name>xe23</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe24</name>
    <config>
      <name>xe24</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe25</name>
    <config>
      <name>xe25</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe26</name>
    <config>
      <name>xe26</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>xe27</name>
    <config>
      <name>xe27</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
  <interface>
    <name>po1</name>
    <config>
      <name>po1</name>
      <shutdown operation="delete"></shutdown>
    </config>
  </interface>
```

```
</interfaces>
```

To commit the configuration

```
commit --rpc-timeout 900
```

To copy running-config to startup-config

```
copy-config --source running --target startup
```

To verify interface counter rate and interface status and other show commands using CMLSH

```
SP-LEAF1#sh int cou r m
+-------------------+--------------+-------------+--------------+-------------+
|     Interface     |    Rx mbps   |    Rx pps   |    Tx mbps   |    Tx pps   |
+-------------------+--------------+-------------+--------------+-------------+
 ce2                  0.37           137           0.57           157
 po100                0.49           51            0.49           51
 po100.101            0.10           9             0.10           9
 po100.102            0.00           0             0.10           9
 po100.103            0.10           9             0.00           0
 po100.104            0.10           9             0.10           9
 po100.107            0.10           9             0.10           9
 po100.108            0.10           9             0.00           0
 po100.109            0.00           0             0.10           9
 xe5                  0.21           26            0.00           5
 xe22                 0.49           51            0.49           51


SP-LEAF1#sh run interface
!
interface po1
!
interface po100
 switchport
 load-interval 30
 evpn multi-homed system-mac aaaa.aaaa.aaa1
!
interface po100.100 switchport
 encapsulation dot1q 100
 rewrite pop
 access-if-evpn
  map vpn-id 100
!
interface po100.101 switchport
 encapsulation dot1q 101
 rewrite pop
 access-if-evpn
  map vpn-id 101
!
interface po100.102 switchport
 encapsulation dot1q 102
 rewrite pop
 access-if-evpn
  map vpn-id 102
!
interface po100.103 switchport
 encapsulation dot1q 103
 rewrite pop
 access-if-evpn
  map vpn-id 103
!
interface po100.104 switchport
```

```
 encapsulation dot1q 104
 rewrite pop
 access-if-evpn
  map vpn-id 104
!
interface po100.105 switchport
 encapsulation dot1q 105
 rewrite pop
 access-if-evpn
  map vpn-id 105
!
interface po100.106 switchport
 encapsulation dot1q 106
 rewrite pop
 access-if-evpn
  map vpn-id 106
!
interface po100.107 switchport
 encapsulation dot1q 107
 rewrite pop
 access-if-evpn
  map vpn-id 107
!
interface po100.108 switchport
 encapsulation dot1q 108
 rewrite pop
 access-if-evpn
  map vpn-id 108
!
interface po100.109 switchport
 encapsulation dot1q 109
 rewrite pop
 access-if-evpn
  map vpn-id 109
!
interface cd0
!
interface cd1
!
interface ce2
 load-interval 30
 ip address 90.90.90.1/24
!
interface ce3
!
interface eth0
 ip vrf forwarding management
 ip address dhcp
!
interface lo
 ip address 127.0.0.1/8
 ip address 1.1.1.1/32 secondary
 ipv6 address ::1/128
!
interface lo.management
 ip vrf forwarding management
 ip address 127.0.0.1/8
 ipv6 address ::1/128
!
interface xe4
!
interface xe5
 speed 10g
 load-interval 30
 ip address 60.60.60.1/24
!
interface xe6
!
```

```
interface xe7
!
interface xe8
!
interface xe9
!
interface xe10
!
interface xe11
!
interface xe12
!
interface xe13
!
interface xe14
!
interface xe15
!
interface xe16
!
interface xe17
!
interface xe18
!
interface xe19
!
interface xe20
!
interface xe21
!
interface xe22
 speed 10g
 channel-group 100 mode active
!
interface xe23
!
interface xe24
!
interface xe25
!
interface xe26
!
interface xe27
!
```

To verify service tracking is in progress

```
get --filter-subtree=/root/Beluganos/IUTD/filters/filter-service-tracking.xml


cat filter-service-tracking.xml
<services-tracking xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-service-tracking"/>

OUTPUT:
> get --filter-subtree=/root/nantha/filters/filter-service-tracking.xml
DATA
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <services-tracking xmlns="http://www.ipinfusion.com/yang/ocnos/ipi-service-tracking">
    <service-tracking>
      <service-type>IF</service-type>
      <state>
        <service-type>IF</service-type>
        <operational-status-check>UP</operational-status-check>
        <status>SUCCESS</status>
      </state>
    </service-tracking>
```

```
  </services-tracking>
</data>
```