



# OcNOS<sup>®</sup>

## Open Compute Network Operating System Version 6.6.0

NetConf User Guide  
February 2025

© 2025 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, California 95054  
<http://www.ipinfusion.com/>

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

[support@ipinfusion.com](mailto:support@ipinfusion.com)

Trademarks:

IP Infusion and OcNOS 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

---

Preface	7
IP Maestro Support	7
Audience	7
Conventions	7
Chapter Organization	7
Related Documentation	7
Feature Availability	8
Migration Guide	8
Support	8
Comments	8
Overview	9
Command Line Interface Help	9
Command Completion	10
Command Abbreviations	10
Command Line Errors	10
Command Negation	11
Syntax Conventions	11
Variable Placeholders	12
Command Description Format	13
Keyboard Operations	13
Show Command Modifiers	14
Begin Modifier	14
Include Modifier	15
Exclude Modifier	15
Redirect Modifier	16
Last Modifier	16
String Parameters	17
Command Modes	17
Command Mode Tree	18
Transaction-based Command-line Interface	19
CHAPTER 1 Introduction	20
CHAPTER 2 NetConf Quick Start	21
NetConf Clients	21
Install Yangcli	21
Download Yang files	21
CHAPTER 3 Yangcli Operations	23
Establish Connection	23
Configure the Device	23
Limitations	26
Retrieve Candidate Configuration	27

Commit Candidate Configuration . . . . .	28
Retrieve Running Configuration . . . . .	28
Retrieve Running Configuration and State Data . . . . .	28
Copy Running Configuration to Startup . . . . .	29
Error Messages . . . . .	29
Warning Messages . . . . .	30
Candidate Configuration store lock . . . . .	31
Startup configuration store lock . . . . .	32
Running configuration store lock . . . . .	33
Force unlock . . . . .	33
Sys-reload . . . . .	34
Sys-shutdown . . . . .	34
<b>CHAPTER 4 Transactions . . . . .</b>	<b>36</b>
Discard Changes . . . . .	36
Commit . . . . .	38
Confirmed-commit . . . . .	38
commit confirmed . . . . .	38
cancel-commit . . . . .	39
commit confirmed persist= <id> . . . . .	40
commit . . . . .	41
Transaction Limit . . . . .	42
<b>CHAPTER 5 NetConf Monitoring . . . . .</b>	<b>44</b>
Retrieve YANG Schema . . . . .	44
Error Messages . . . . .	44
Retrieving Schema List . . . . .	45
<b>CHAPTER 6 URL Capabilities . . . . .</b>	<b>46</b>
Overview . . . . .	46
Limitations . . . . .	46
Prerequisites . . . . .	46
Copy-config . . . . .	46
From HTTP to the Candidate Configuration Store . . . . .	46
From HTTPs to the Candidate Configuration Store . . . . .	47
From FTP to the Candidate Configuration Store . . . . .	48
From SFTP to the Candidate Configuration Store . . . . .	49
From a Local file to the Candidate Configuration Store . . . . .	50
From the Candidate Configuration Store to HTTP . . . . .	50
From the Candidate Configuration Store to HTTPS . . . . .	51
From the Candidate Configuration Store to FTP . . . . .	51
From the Candidate Configuration Store to SFTP . . . . .	52
From the Candidate Configuration Store to the Local File . . . . .	52
Copy-config Error Messages . . . . .	53
Edit-config . . . . .	53
Edit-config using HTTP . . . . .	54
Edit-config using FTP . . . . .	55
Edit-config using local file . . . . .	55

---

Edit-config Error Messages . . . . .	56
Delete-config . . . . .	56
Delete configuration from HTTP server . . . . .	56
Delete configuration from FTP . . . . .	57
Delete local file . . . . .	57
Delete-config Error Messages . . . . .	58
CHAPTER 7 Event Notification . . . . .	60
Client Notification Subscription . . . . .	60
netconf-config-change . . . . .	60
Notes . . . . .	61
Example . . . . .	61
netconf-capability-change . . . . .	62
netconf-session-start . . . . .	62
netconf-session-end . . . . .	62
CHAPTER 8 Validate Capability . . . . .	64
CHAPTER 9 With-defaults Capability . . . . .	65
Explicit Mode . . . . .	65
Trim Mode . . . . .	65
Report-All Mode . . . . .	65
Report-All-Tagged Mode . . . . .	65
Edit-config Behavior . . . . .	66
CHAPTER 10 Supported Operations . . . . .	67
CHAPTER 11 Sys-Update using NetConf . . . . .	71
Download the Image and Install . . . . .	71
Validation . . . . .	71
To Delete the Downloaded Image . . . . .	72
Validation . . . . .	72
To Cancel the Image Download: . . . . .	72
Validation . . . . .	72
CHAPTER 12 SSH Client . . . . .	74
Establish a Connection . . . . .	74
Send Client Help Message to NetConf Server . . . . .	74
NETCONF-SSH over User Defined VRF . . . . .	75
Server Configuration for User Defined VRFs . . . . .	76
Server Configuration for User Defined VRFs to Configure SSH Ports . . . . .	76
Validation . . . . .	76
CHAPTER 13 OpenDaylight Controller . . . . .	77
Documentation . . . . .	77
Installation . . . . .	77
Connecting with an OcNOS Device . . . . .	77
Restconf Endpoints . . . . .	78
Patch or Merge Operation . . . . .	79
POSTMAN . . . . .	82
Installation . . . . .	82

---

---

Edit-config .....	83
Edit-config using API-DOCS .....	83
Edit-config using Terminal .....	84
Edit-config using POSTMAN .....	84
Known Issues and Troubleshooting .....	85
<b>CHAPTER 14 NetConf Over Transport Layer Security .....</b>	<b>88</b>
Topology .....	88
Client Configuration .....	88
Server Configuration .....	89
Validation .....	89
NETCONF-TLS over User Defined Vrfs .....	90
Validation .....	90
<b>CHAPTER 15 Call Home .....</b>	<b>92</b>
Configuration .....	92

---

# Preface

---

This guide describes how to configure OcNOS.

---

## IP Maestro Support

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

---

## Audience

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

---

## Conventions

[Table 1](#) on page 7 shows the conventions used in this guide.

**Table 1: Conventions**

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

---

## Chapter Organization

The chapters in command references are organized as described in [Command Description Format](#).

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

---

## Related Documentation

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

---

## Feature Availability

The features described in this document that are available depend upon the OcNOS SKU that you purchased. See the *Feature Matrix* for a description of the OcNOS SKUs.

---

## Migration Guide

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

---

## Support

For support-related questions, contact [support@ipinfusion.com](mailto:support@ipinfusion.com).

---

## Comments

If you have comments, or need to report a problem with the content, contact [techpubs@ipinfusion.com](mailto:techpubs@ipinfusion.com).



---

# 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.

---

## 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
```

---

## 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 authent
ication-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](#) on page 11 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	<code>show ip ospf</code>
lowercase	Keywords that you enter exactly as shown in the command syntax.	<code>show ip ospf</code>
UPPERCASE	See <a href="#">Variable Placeholders</a>	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.	<code>(A.B.C.D &lt;0-4294967295&gt;)</code>
( )	Optional parameters, from which you select one or none. Vertical bars delimit the selections. Do not enter the parentheses or vertical bars as part of the command.	<code>(A.B.C.D &lt;0-4294967295&gt; )</code>
( )	Optional parameter which you can specify or omit. Do not enter the parentheses or vertical bar as part of the command.	<code>(IFNAME )</code>
{ }	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.	<code>{intra-area &lt;1-255&gt; inter-area &lt;1-255&gt; external &lt;1-255&gt;}</code>

**Table 2: Syntax conventions (Continued)**

Convention	Description	Example
[ ]	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](#) on page 12 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
XX:XX:XX:XX:XX:XX	MAC address
<1-5> <1-65535> <0-2147483647> <0-4294967295>	Numeric range

---

## Command Description Format

[Table 4](#) on page 13 explains the sections used to describe each command in this reference.

**Table 4: Command descriptions**

Section	Description
<b>Command Name</b>	The name of the command, followed by what the command does and when should it be used
<b>Command Syntax</b>	The syntax of the command
<b>Parameters</b>	Parameters and options for the command
<b>Default</b>	The state before the command is executed
<b>Command Mode</b>	The mode in which the command runs; see <a href="#">Command Modes</a>
<b>Example</b>	An example of the command being executed

---

## Keyboard Operations

[Table 5](#) on page 13 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
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

**Table 5: Keyboard operations (Continued)**

Key combination	Operation
Ctrl+t	Transposes the current character with the previous character
Ctrl+c	Ignores the current line and redisplay 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

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](#) on page 17 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 <code>description</code> command in interface mode.

---

## Command Modes

Commands are grouped into modes arranged in a hierarchy. Each mode has its own set of commands. [Table P-7](#) lists the command modes common to all protocols.

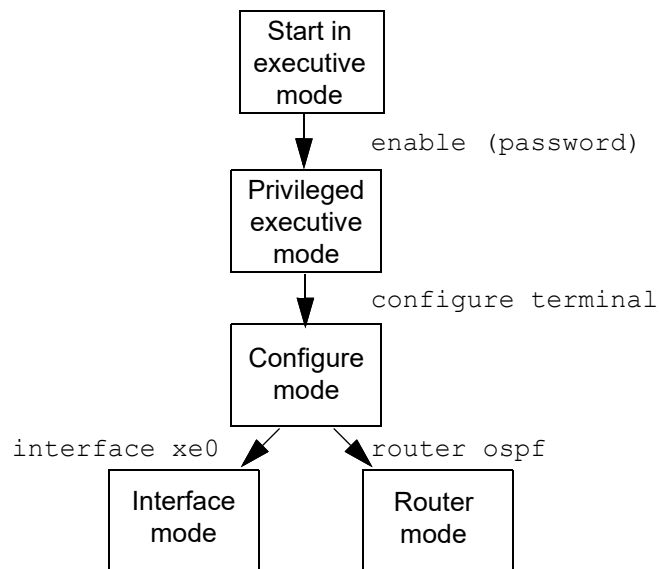
**Table 7: Common command modes**

Name	Description
Executive mode	Also called <i>view</i> 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 <code>show</code> , <code>exit</code> , <code>quit</code> , <code>help</code> , and <code>enable</code> .
Privileged executive mode	Also called <i>enable</i> mode, in this mode you can run additional basic commands such as <code>debug</code> , <code>write</code> , and <code>show</code> .
Configure mode	Also called <i>configure terminal</i> mode, in this mode you can run configuration commands and go into other modes such as <code>interface</code> , <code>router</code> , <code>route map</code> , <code>key chain</code> , and <code>address family</code> .  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.

---

## Command Mode Tree

The diagram below shows the common command mode hierarchy.



**Figure P-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 `commit` 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 `commit`. 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.

## CHAPTER 1 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 [Chapter 4, Transactions](#).

---

## CHAPTER 2 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 at:

[https://github.com/OpenClovis/OpenYuma/blob/master/netconf/doc/yuma\\_docs/openyuma-yangcli-manual.odt](https://github.com/OpenClovis/OpenYuma/blob/master/netconf/doc/yuma_docs/openyuma-yangcli-manual.odt)

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.

Note: 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 [Chapter 12, SSH Client](#).

---

### 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.

---

## CHAPTER 3 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.

**Note:** 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-sessions` 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.

---

### 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 NetConf Command Reference guide.

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 3-1: Edit-config operations

		Default operation		
		Merge	Replace	None
Create	Object	Y	Y	
	Leaf			
Delete	Object	Y	Y	
	Leaf	Y	Y	
Merge	Object	Y	Y	
	Leaf	Y	Y	
Replace	Object	Y	Y	
	Leaf	Y	Y	
Remove	Object	Y	Y	
	Leaf	Y	Y	
<b>Legend:</b>	Y	Supported		
	Blank	Not supported; error will be returned		

## 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
  }
}
}
}
}
}
}
}

```

---

## 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 /bgp rpc-reply {
  data {
    bgp {
      bgp-instance 100 {
        bgp-as 100
        config {
          bgp-as 100
        }
        state {
          bgp-as 100
          scan-remain-time 8
          router-run-time-ip-address 0.0.0.0
          total-prefixes 0
          table-version 1
          version 4
        }
        timers {
          config {
            hold-time 9
            keep-alive 3
          }
          state {
            hold-time 9
            keep-alive 3
          }
        }
      }
    }
  }
}
```

---

## Copy Running Configuration to Startup

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

The equivalent OcNOS command is:

```
# write
```

---

## 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>cel/1</name>
    <config>
      <name>cel/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/ospfv2_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.

---

## 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'  
  }  
}
```

---

## 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
```

---

## CHAPTER 4 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'  
}
```

---

## CHAPTER 5 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
  }
}
}

```

---

## 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>
        ...
        ...
      }
    }
  }
}
}

```

---

## CHAPTER 6 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.

---

### 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 `https 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> 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.

```
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
    }
  }
}
```

```
}  
}  
}
```

---

## CHAPTER 7 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.

---

### Client Notification Subscription

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

Example:

```
yangcli root@127.1> create-subscription
RPC OK Reply 1 for session 1:
yangcli root@127.1>
```

---

### 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
  - 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)

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

- Create Yangcli session

- 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.

- 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>
```

- Do some configuration

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

- 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
  }
}
```

```
    }  
  }  
  
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  
  }  
}
```

---

## CHAPTER 8 Validate Capability

---

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

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:
```



---

## CHAPTER 9 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&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>
</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.

## CHAPTER 10 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 10-2: 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	

Table 10-2: Supported Operations (Continued)

Capability	Operation	User Role	Supported (Yes/No)	Comments
:base:1.0	<edit-config>: <delete>	Operator, Engineer, Admin	Yes	<p>Supports attribute-level 'delete' operation only with 'merge' as default-operation.</p> <p>Note: For a leaf-level delete operation, a value is not required. If given, it is ignored by the server. For example:</p> <pre>&lt;mtu operation="delete"/&gt; &lt;&lt; This is enough to delete/unset a leaf. &lt;mtu operation="delete"&gt;1560&lt;/mtu&gt; &lt;&lt; while processing this, the value (1560) is ignored by server.</pre> <p>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):</p> <pre>&lt;list operation="delete"&gt;   &lt;key&gt;value&lt;/key&gt; &lt;/list&gt;</pre>
:base:1.0	<edit-config>: <remove>	Operator, Engineer, Admin	Yes	<p>Supports attribute-level 'remove' operation only with 'merge' or 'replace'.</p> <p>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.</p> <p>For a leaf-level "remove" operation, a value is not required. If given, it is ignored by the server. For example:</p> <pre>&lt;mtu operation="remove"/&gt; &lt;&lt; This is enough to remove a leaf. &lt;mtu operation="remove"&gt;1560&lt;/mtu&gt; &lt;&lt; while processing this, the value (1560) is ignored by server.</pre> <p>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):</p> <pre>&lt;list operation="remove"&gt;   &lt;key&gt;value&lt;/key&gt; &lt;/list&gt;</pre>

Table 10-2: Supported Operations (Continued)

Capability	Operation	User Role	Supported (Yes/No)	Comments
: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	
: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	

**Table 10-2: Supported Operations (Continued)**

<b>Capability</b>	<b>Operation</b>	<b>User Role</b>	<b>Supported (Yes/No)</b>	<b>Comments</b>
: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.

## CHAPTER 11 Sys-Update using NetConf

This chapter 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.

### Download the Image and Install

#### On the Switch:

Connect to `yangcli` and proceed as shown below:

<pre>yangcli ocnos@127.1&gt; 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</pre>	<p>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)</p>
<pre>yangcli ocnos@127.1&gt; sys-update-install installerName=EC_AS7326_56X-OcNOS- 1.3.6.241a-DC_IPBASE-S0-P0-installer</pre>	<p>Install the downloaded image.</p>

**Note:** 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:

<pre>yangcli ocnos@127.1&gt; 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</pre>	<p>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)</p>
<pre>yangcli ocnos@127.1&gt; sys-update-delete imageName=EC_AS7326_56X-OcNOS-1.3.6.241a- DC_IPBASE-S0-P0-installer</pre>	<p>Delete the downloaded image.</p>

---

## 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:

<pre>yangcli ocnos@127.1&gt; 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</pre>	<p>To download the OcNOS image.</p>
<pre>yangcli ocnos@127.1&gt; sys-update-cancel- download</pre>	<p>Cancel the download.</p>

---

## 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  
#
```

---

## CHAPTER 12 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>
          <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
```

---

## CHAPTER 13 OpenDaylight Controller

---

---

### Documentation

The Opendaylight-0.12.2 user guide 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.

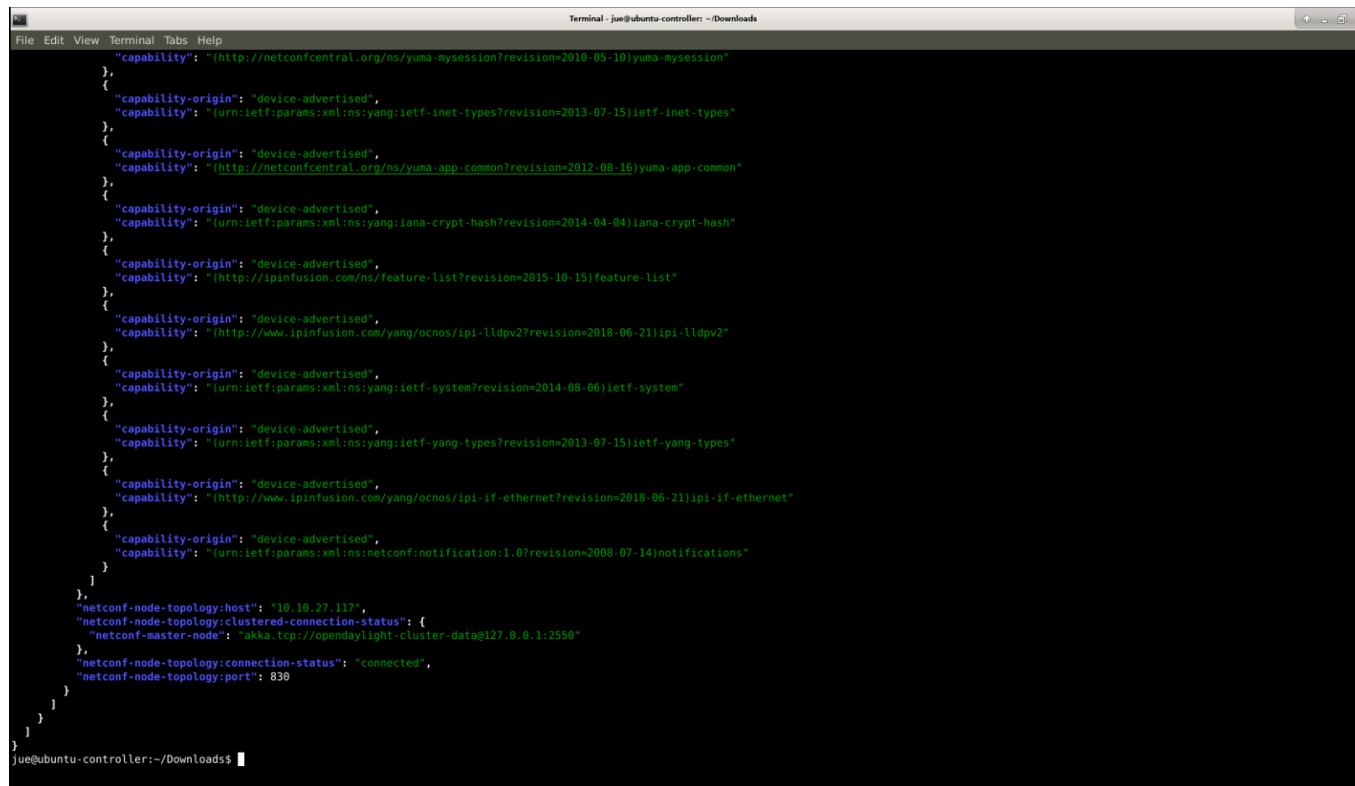
```
curl --user "admin":"admin" -H "Content-type: application/xml" -X PUT http://localhost:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/ocnos-3 -d '@ocnos_connect.xml'
```

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

```
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:



```
Terminal - jue@ubuntu-controller: ~/Downloads
File Edit View Terminal Tabs Help
{
  "capability": "http://netconfcentral.org/ns/yuma-mysession?revision=2010-05-10)yuma-mysession",
  {
    "capability-origin": "device-advertised",
    "capability": "urn:ietf:params:xml:ns:yang:ietf-inet-types?revision=2013-07-15)ietf-inet-types",
    {
      "capability-origin": "device-advertised",
      "capability": "http://netconfcentral.org/ns/yuma-app-common?revision=2012-08-16)yuma-app-common",
      {
        "capability-origin": "device-advertised",
        "capability": "urn:ietf:params:xml:ns:yang:iana-crypt-hash?revision=2014-04-04)iana-crypt-hash",
        {
          "capability-origin": "device-advertised",
          "capability": "http://ipinfusion.com/ns/feature-list?revision=2015-10-15)feature-list",
          {
            "capability-origin": "device-advertised",
            "capability": "http://www.ipinfusion.com/yang/ocnos/ipi-lldpv2?revision=2018-06-21)ipi-lldpv2",
            {
              "capability-origin": "device-advertised",
              "capability": "urn:ietf:params:xml:ns:yang:ietf-system?revision=2014-08-06)ietf-system",
              {
                "capability-origin": "device-advertised",
                "capability": "urn:ietf:params:xml:ns:yang:ietf-yang-types?revision=2013-07-15)ietf-yang-types",
                {
                  "capability-origin": "device-advertised",
                  "capability": "http://www.ipinfusion.com/yang/ocnos/ipi-if-ethernet?revision=2018-06-21)ipi-if-ethernet",
                  {
                    "capability-origin": "device-advertised",
                    "capability": "urn:ietf:params:xml:ns:netconf:notification:1.0?revision=2008-07-14)notifications",
                    {
                      "netconf-node-topology:host": "10.10.27.117",
                      "netconf-node-topology:clustered-connection-status": {
                        "netconf-master-node": "akka.tcp://.opendaylight-cluster-data@127.0.0.1:2550"
                      },
                      "netconf-node-topology:connection-status": "connected",
                      "netconf-node-topology:port": 830
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
jue@ubuntu-controller:~/Downloads$
```

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/...](http://localhost:8181/restconf/)
- RFC 8040, [http://localhost:8181/rests/...](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.

```

opendaylight-user@root>feature:info odl-restconf
Feature odl-restconf 1.11.2
Description:
  OpenDaylight :: Restconf
Details:
  OpenDaylight is leading the transformation to Open Software Defined Networking (SDN). For more information, please see https://www.opendaylight.org
Feature has no configuration
Feature has no configuration files
Feature depends on:
  odl-restconf-nb-bierman02 1.11.2
  odl-restconf-nb-rfc8040 1.11.2
Feature has no bundles.
Feature has no conditionals.
opendaylight-user@root>

```

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"
      }
    }
  ]
}

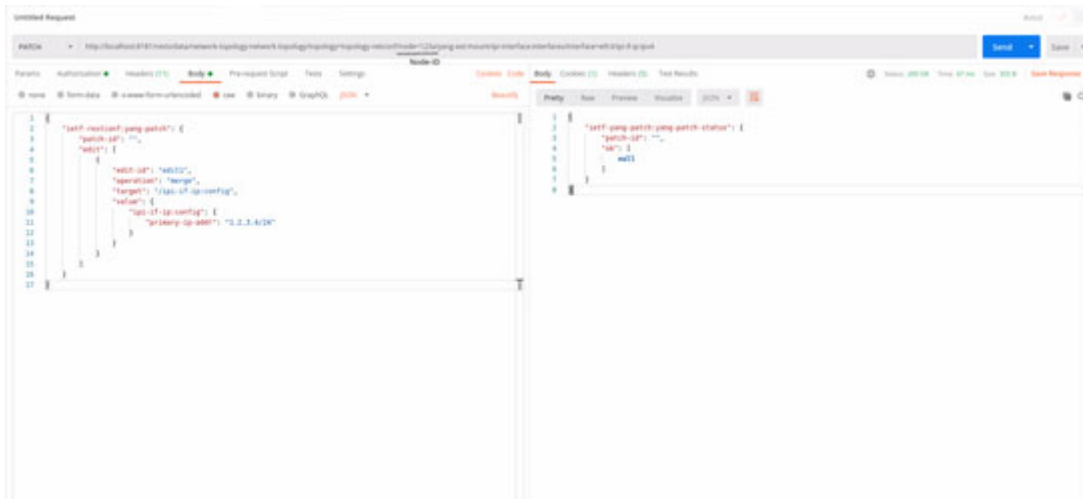
```

#### 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>

```



#### Header for the action:

PATCH		http://localhost:8181/rests/data/network-topology:network-topology/
Params	Authorization ●	<b>Headers (11)</b>
<input checked="" type="checkbox"/> Hide auto-generated headers		Body ● Pre-request Script T
KEY	VALUE	
<input checked="" type="checkbox"/> Authorization ⓘ	Basic YWRtaW46YWRtaW4=	
<input checked="" type="checkbox"/> Cookie ⓘ	JSESSIONID=node014foc8ktwbp71qwig959qp2xx5.nod...	
<input checked="" type="checkbox"/> Postman-Token ⓘ	<calculated when request is sent>	
<input type="checkbox"/> Content-Type ⓘ	application/json	
<input checked="" type="checkbox"/> Content-Length ⓘ	<calculated when request is sent>	
<input checked="" type="checkbox"/> Host ⓘ	<calculated when request is sent>	
<input checked="" type="checkbox"/> User-Agent ⓘ	PostmanRuntime/7.26.8	
<input checked="" type="checkbox"/> Accept ⓘ	*/*	
<input checked="" type="checkbox"/> Accept-Encoding ⓘ	gzip, deflate, br	
<input checked="" type="checkbox"/> Connection ⓘ	keep-alive	
<input checked="" type="checkbox"/> Content-Type	application/yang.patch+json	
Key	Value	

## 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>

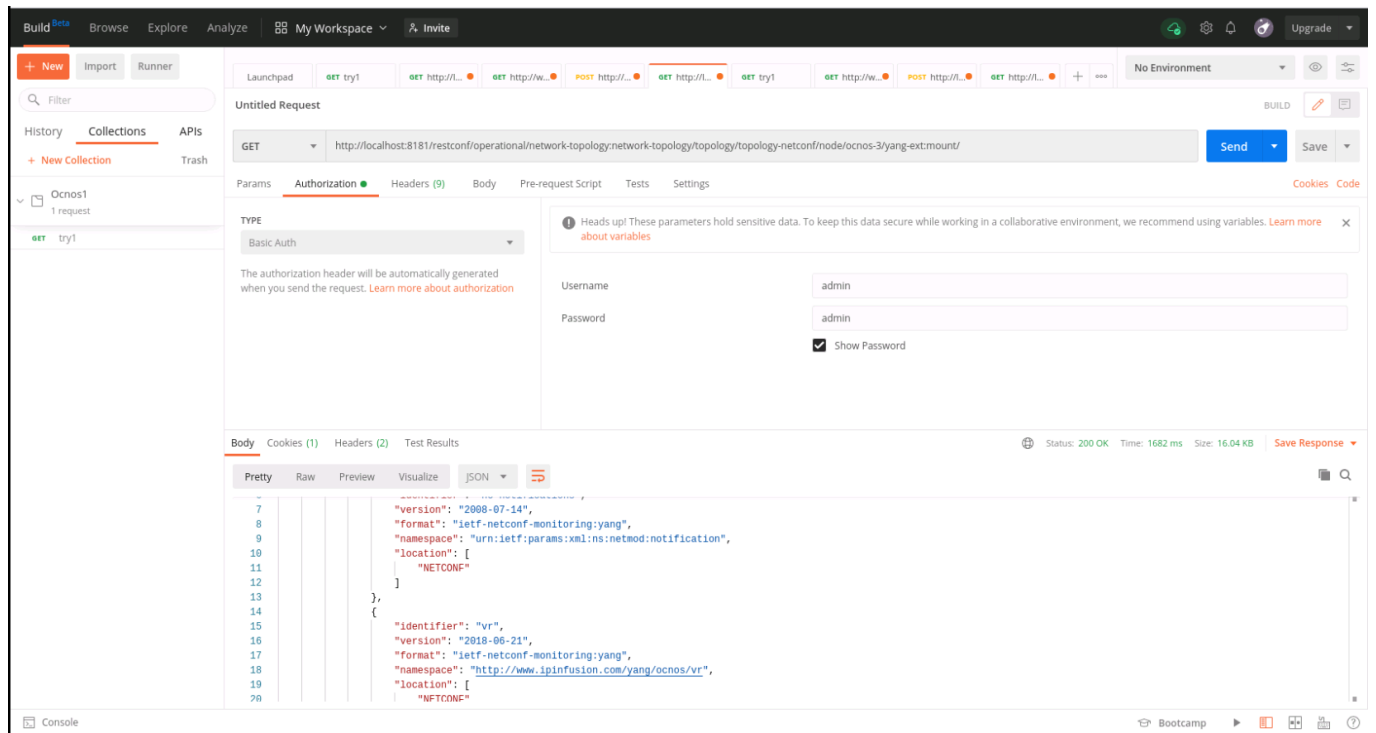
The POSTMAN desktop version should be running in the system while the web version is accessed at this link:

<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.

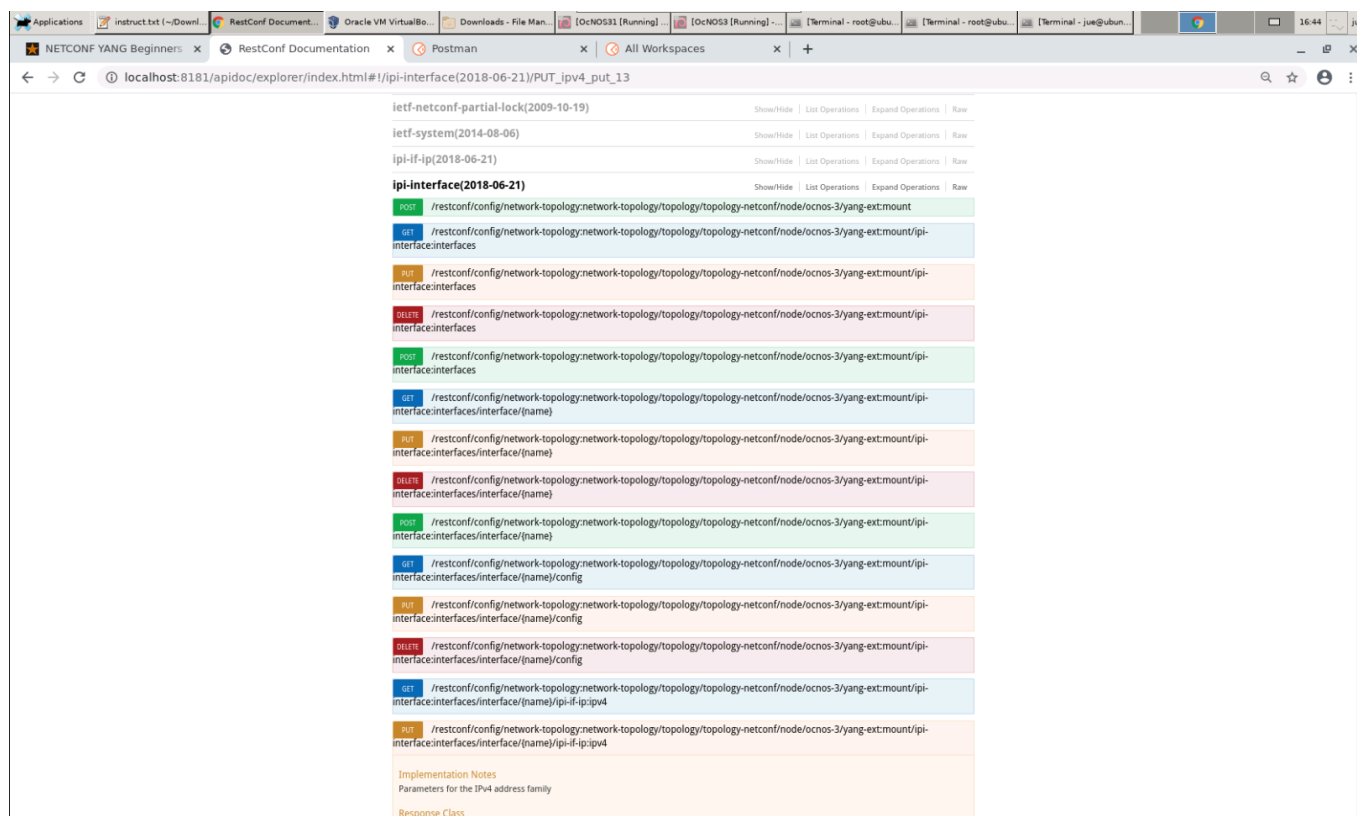
Configuration using POSTMAN is shown later in the section "edit-config".



## Edit-config

### Edit-config using API-DOCS

Under the mounted resources of API-DOCS, all the APIs show the possible RESTCONF 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

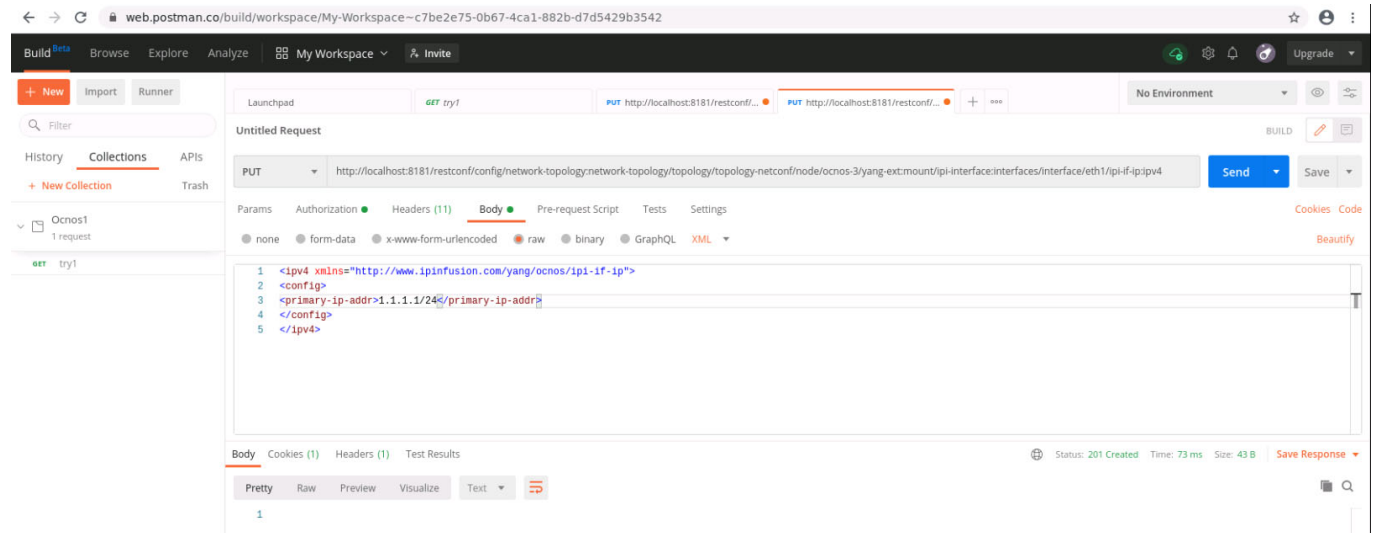
```
$ curl -user "admin":"admin" -H "Content-type: application/xml" -X PUT http://localhost:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/ocnos3/yang-ext:mount/ipi-interface:interfaces/interface/eth1/ipi-if-ip:ipv4 -d '@interface_config.xml'
```

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>
```

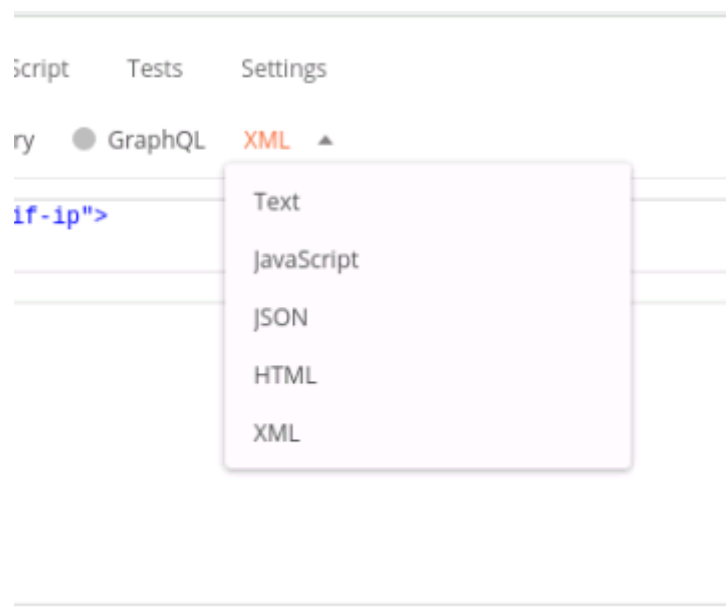
## Edit-config using POSTMAN

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-amd64/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 devices with different product versions cannot connect simultaneously

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.

## CHAPTER 14 NetConf Over Transport Layer Security

Transport Layer Security (TLS) is a cryptographic protocol that uses mutual certificate-based authentication and provides a secure and reliable connection between two devices. It is a successor to the Secure Sockets Layer (SSL) protocol. When a NetConf session is established over TLS, the NetConf server acts as the TLS server, and the NetConf client must act as the TLS client.

NetConf sessions over TLS provide some advantages over sessions that use SSH. Whereas SSH authenticates a client by using credentials (username and password) or keys, TLS uses certificates to mutually authenticate both the client and the server. Certificates can provide additional information about a client and can be used to securely authenticate one device to another. Thus, while NetConf sessions over SSH work well for manually managing individual devices, NetConf sessions that use TLS enable secure device-to-device communication for more effectively managing and automating devices in large-scale networks.

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`.

The NetConf server uses TCP port number 6513 to listen for TCP connections established by NetConf over TLS clients.

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

### Topology



Figure 14-1: Netconf over TLS Topology

### Client Configuration

1. Generate CA authority key and certificate on TLS client:

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

2. Generate client key and certificate on client:

- a. Create a new file named `ClientCertReq.config` with this content:

```
[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
```

```
openssl req -newkey rsa:2048 -keyout ClientKey.pem -out ClientCert.csr -config ./
ClientCertReq.config -nodes -days 100
```

**3. Client certificate signing:**

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

**4. Server certificate signing:**

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

**5. 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 cert.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).

**6. Establish TLS session using the below command on the client side:**

```
connect --tls --host 10.12.89.152 --port 6513 --cert /root/QUX/ClientCert.pem
--key /root/QUX/ClientKey.pem --trusted /root/QUX/rootCACert.pem
```

Note: This example uses the Netopeer2 command as NetConf client over TLS.

## Server Configuration

Generate server private key and CSR request on the TLS server:

>enable	Turn on privileged mode.
#crypto pki generate rsa common-name ipv4 10.12.93.111	IP address used by TLS clients to connect to the NetConf server running on Ocnos.
#show crypto csr	Show Certificate Signing Request

## Validation

On TLS server:

```
# show crypto csr
-----BEGIN CERTIFICATE REQUEST-----
MIICXDCCAUAQCAQAwFzEVMBMGAlUEAwMMTAUuMTIuOTMuMTEExMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEAE7dTLZpS7sTCrFdP4gQD6X4+PHtYKK7jQdSaM
WaX20dkPqxN8/6VHuY3+fe13eeUSPrDbdSOzPq6f/Rdd/xG76Qi95yYKZnUkTz1/
sF50LfdxL2u+Xg25TA4+Lh6EHYxbmLwX6/o4b8E+78CH+NftH6g1/2YN14PC31I1
Jlar+YEAzlyy8q1RAHbiazyJmgR0n4KxdtU/c/nzU8zP40WtJdNquAAMkQUcWwBg
0a88FpFem5SkQRs1LX2Bdofp00SCiMJVkJYR8yirxKwHx7JBg7EtDovofY585rj
zBZBTyCSWYrpWgh+YUm+5ZbEHN+NC2r9UqvLVuHyYWRPB40jxQIDAQABoAAwDQYJ
KoZiHvcNAQELBQADggEBAD0/4MxtfJrz3jeBAMUwIjTBFauTI2rJ0AMxjIcPfel
wLi/nK4HBU3Ucg9yUqfPYwfi3gLa901AJT5UVPZV0C783nNBrR9GRa91rL0+0Ksa
hptlTCvFug/N7oEiADQ2IHskNTyCSW7MeJaz06amhiGHp+QNjNulAmsXUjPOzKsJ
```

```
OILOmBvD10N+GvtvBMhJrxDKzCda9auSdk1If1BRdfmNttnfthoK3PWK5kmkyh1r
14mz3Mvd9E5UvsMfL8u72FibwaHa3b862/YQdbidN2LS8xjCZTxeitbJooLzywhb
hR99BENUG8zTbmIa8BD2Nt8F3mlo/31kD8eO2QTLqfI=
-----END CERTIFICATE REQUEST-----
PE3#
```

**Note:** You should copy and paste the output of the above show command into the `ServerCert.csr` file on the client.

---

## 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"
(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

---

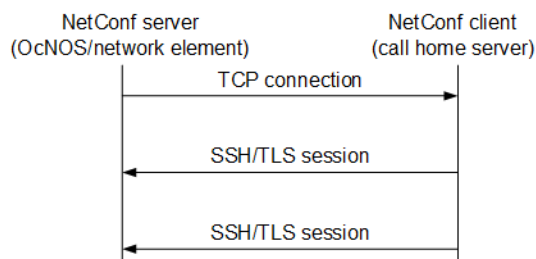
## Validation

```
#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
```

## CHAPTER 15 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 15-2](#), this process is standardized by IETF in RFC 8071.



**Figure 15-2: 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*.