# OcNOS®

## Open Compute
## Network Operating System
## Version 6.4.2

# NetConf User Guide

December 2023

IP Infusion Inc.
3965 Freedom Circle, Suite 200
Santa Clara, CA 95054
+1 408-400-1900
http://www.ipinfusion.com/

For support, questions, or comments via E-mail, contact:
support@ipinfusion.com

# Contents

# Preface

This guide describes how to configure NetConf in OcNOS.

## Audience

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

## Conventions

Table P-1 shows the conventions used in this guide.

**Table P-1: Conventions**

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

## Support

For support-related questions, contact support@ipinfusion.com.

## Comments

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

# CHAPTER 1 Introduction

This document describes managing OcNOS devices using NetConf.

This document is intended for network administrators and other 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 through which a network device can be managed, configuration data information can be retrieved, and new configuration data can be uploaded.

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 in all sessions. The candidate and running and startup configuration are shared across all the 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. In this document, the OpenYuma Yangcli application is used 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

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 you are 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 that is 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.

## 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, it is necessary to set other non-key attributes of the referenced interface list. 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"`).

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

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

```
yangcli ocnos@127.1> create-subscription

RPC OK Reply 3 for session 1:
```

Execute the following command and commit the changes:

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

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

RPC OK Reply 6 for session 1:


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

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

## Candidate Configuration store lock

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

NetConf Client 1: locks candidate configuration store.

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

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

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

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

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

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

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

NetConf client 1 releases the lock

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

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

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

## Startup configuration store lock

NetConf client 1 locks startup configuration store

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

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

## Running configuration store lock

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

## 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 exists.

```
<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

OcNOS supports: url capability. The URL capability is identified by the following capability string:

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

OcNOS supports HTTP, FTP, and file schemes under this capability for these operations.

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

## Limitations

- Operation copy-config operation with source as URL and target as startup is not supported.
- Operation copy-config with source and target with different URL is not supported.
- Operation copy-config with source as URL and target as running is not supported.
- Operation copy-config with source and target with same URL is not a valid operation.

## <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"`

### Copy URL configuration to candidate configuration store

As mentioned above, HTTP, FTP and file schemes are supported in URL capability.

Note: Yangcli supports URL option only in interactive mode.

### HTTP to Candidate configuration store

Download configuration from HTTP server and copy it to 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
```

### FTP to Candidate configuration store

Download configuration from FTP server and copy it to 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
```

## Local file to Candidate configuration store

Using file scheme support, user shall copy the configuration from stored local file to 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:
```

### Copy candidate configuration store to URL

Herein, Candidate configuration store is chosen to show the procedural steps.

Note: Source configuration store can also be chosen either Running or Startup.

### Candidate configuration store to HTTP

Upload configuration to 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 URL with basic authentication.

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

### 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
```

### Candidate configuration store to local file

Using file scheme support, user shall create a snapshot of configuration store. Here is the example to copy URL configuration store into a local file.

Note: Local files are created under `/root/.yuma/` directory of the device; support to store 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 present in 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)
2. Enable config change notification (command: config-change-subscription status=enable)
3. Perform any of the following operations
   - Create
   - Merge
   - Delete
   - Replace
4. Do commit
5. Config change notification will be received.

6. Config change notification can be disabled for current session. (command: config-change-subscription status=disable)
7. Someone can check whether current session has been subscribed for config change notificaion. (command: show-config-change-subscription)

## Example

1. Create Yangcli session
2. Subscribe for notification

```
yangcli root@0> create-subscription
RPC OK Reply 4 for session 1
```

3. Subscribe for config change notification

```
yangcli root@0> config-change-subscription status=enable
RPC OK Reply 5 for session 1
```

4. Someone can check whether current session has been subscribed for config change notificaion

```
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>
```

5. Do some configuration

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

6. 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&amp;also-supported=trim,report-all,report-all-tagged</nc:capability>
```

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

## Explicit Mode

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

## Trim Mode

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

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

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

## Report-All Mode

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

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

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

## Report-All-Tagged Mode

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

```
<get-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <source>
    <running/>
  </source>
  <with-defaults xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">report-all-tagged</with-defaults>
</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 | Supports attribute-level 'delete' operation only with 'merge' as default-operation.<br><br>Note: For a leaf-level delete operation, a value is not required. If given, it is ignored by the server. For example:<br><br><mtu operation="delete"/> << This is enough to delete/unset a leaf.<br><mtu operation="delete">1560</mtu><br><< while processing this, the value (1560) is ignored by server.<br><br>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):<br><br><list operation="delete"><br><key>value</key><br></list> |
| :base:1.0 | <edit-config>: <remove> | Operator, Engineer, Admin | Yes | Supports attribute-level 'remove' operation only with 'merge' or ‘replace’.<br><br>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.<br><br>For a leaf-level “remove” operation, a value is not required. If given, it is ignored by the server. For example:<br><br><mtu operation=”remove”/> << This is enough to remove a leaf.<br><mtu operation="remove">1560</mtu><br><< while processing this, the value (1560) is ignored by server.<br><br>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):<br><br><list operation="remove"><br><key>value</key><br></list> |

**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)**

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

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

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:

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

## Validation

Show installer output after completion of image download:

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

Show installer output after deleting downloaded image:

```
#show installers
#
```

## To Cancel the Image Download:

### On the Switch:

Connect to yangcli and proceed as shown below:

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

## 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>]]>]]>
```

# 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.

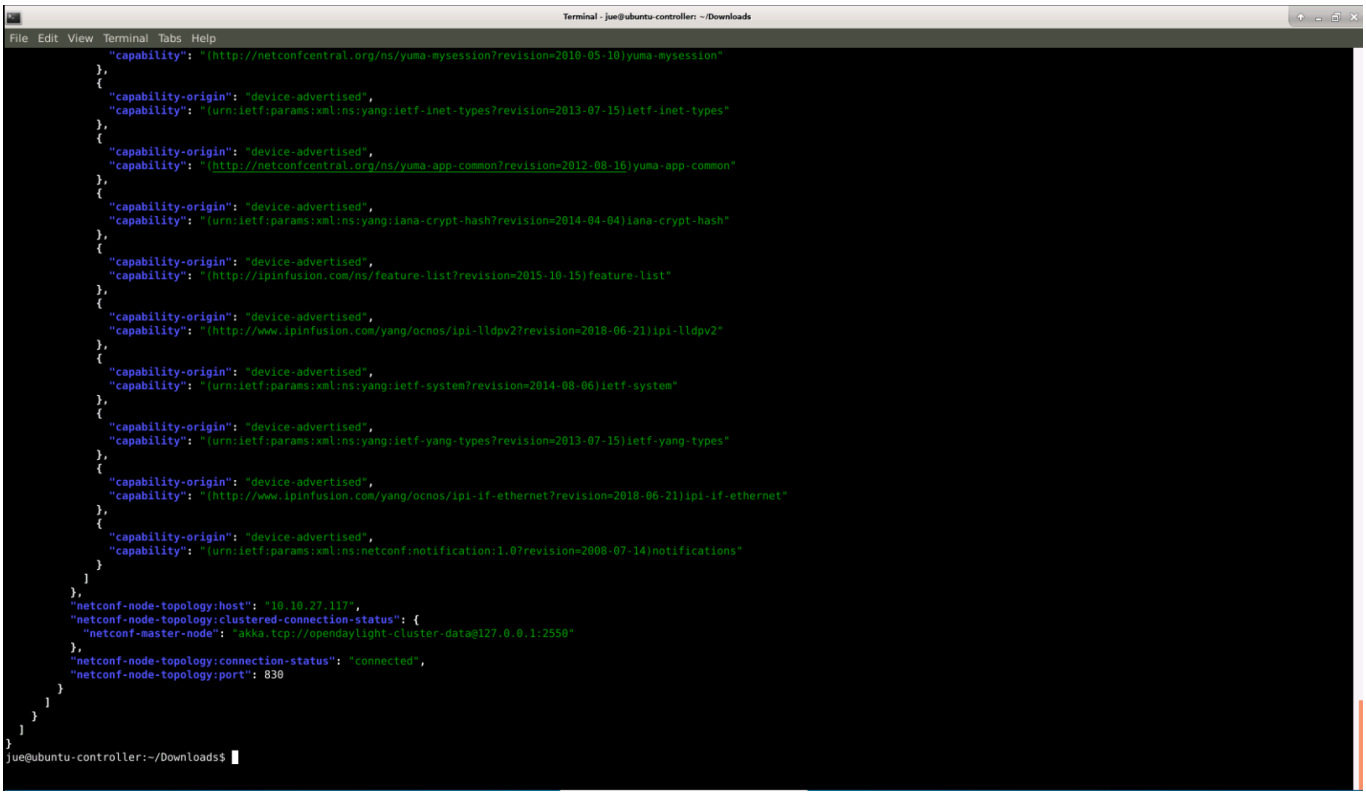
```
curl --user "admin":"admin" -H "Content-type: application/xml" -X PUT http://
localhost:8181/restconf/config/network-topology:network-topology/topology/topology-
netconf/node/ocnos-3 -d '@ocnos_connect.xml'
```

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:




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

## Restconf Endpoints

There are 2 different endpoints related to RestConf protocols:

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

  For example:

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

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

```
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
```

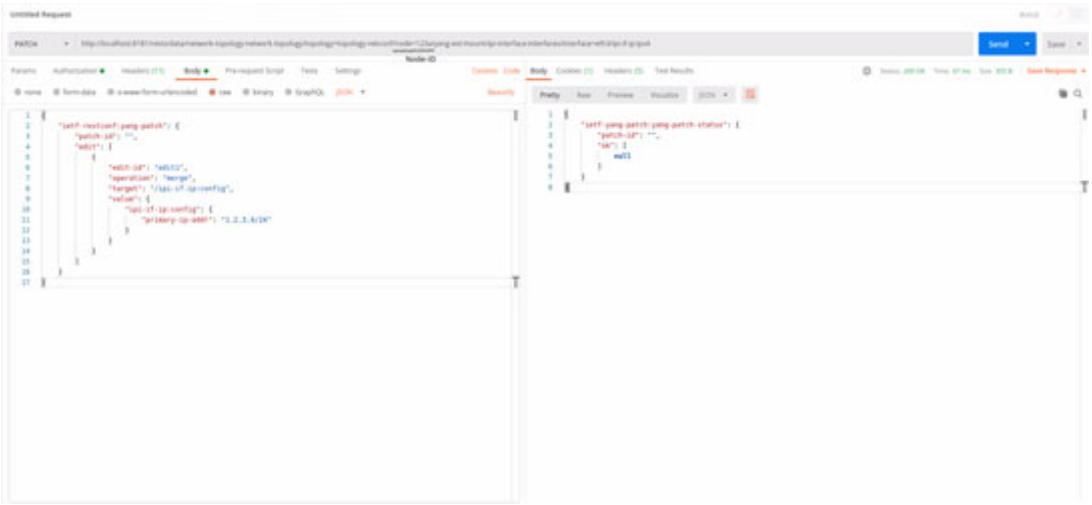
JSON payload:

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

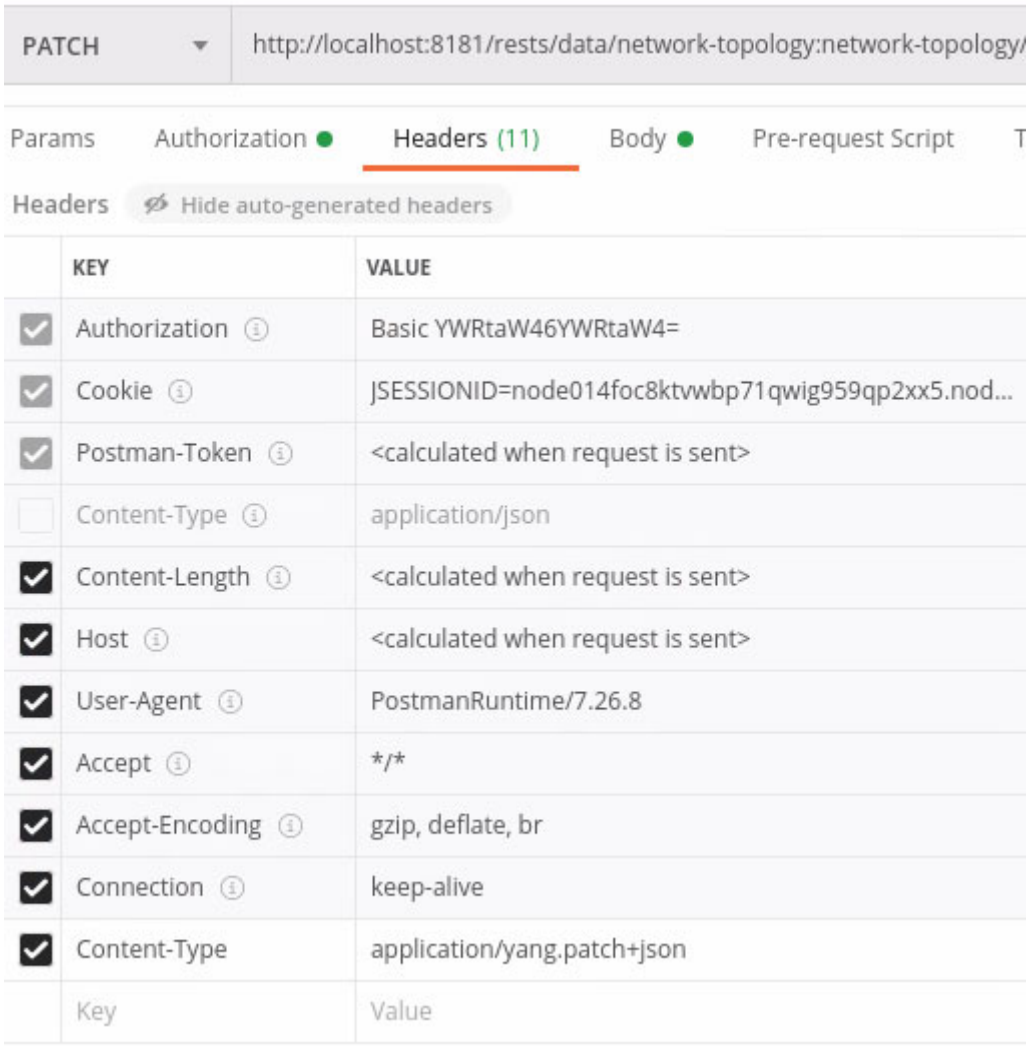
```
                "value": {
                    "ipi-if-ip:config": {
                        "primary-ip-addr": "x.x.x.x/24"
                    }
                }
            }
        ]
    }
}
```

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:




# POSTMAN

## Installation

```
$ sudo snap install postman
```

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

For more details for downloading and installing POSTMAN desktop version:

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

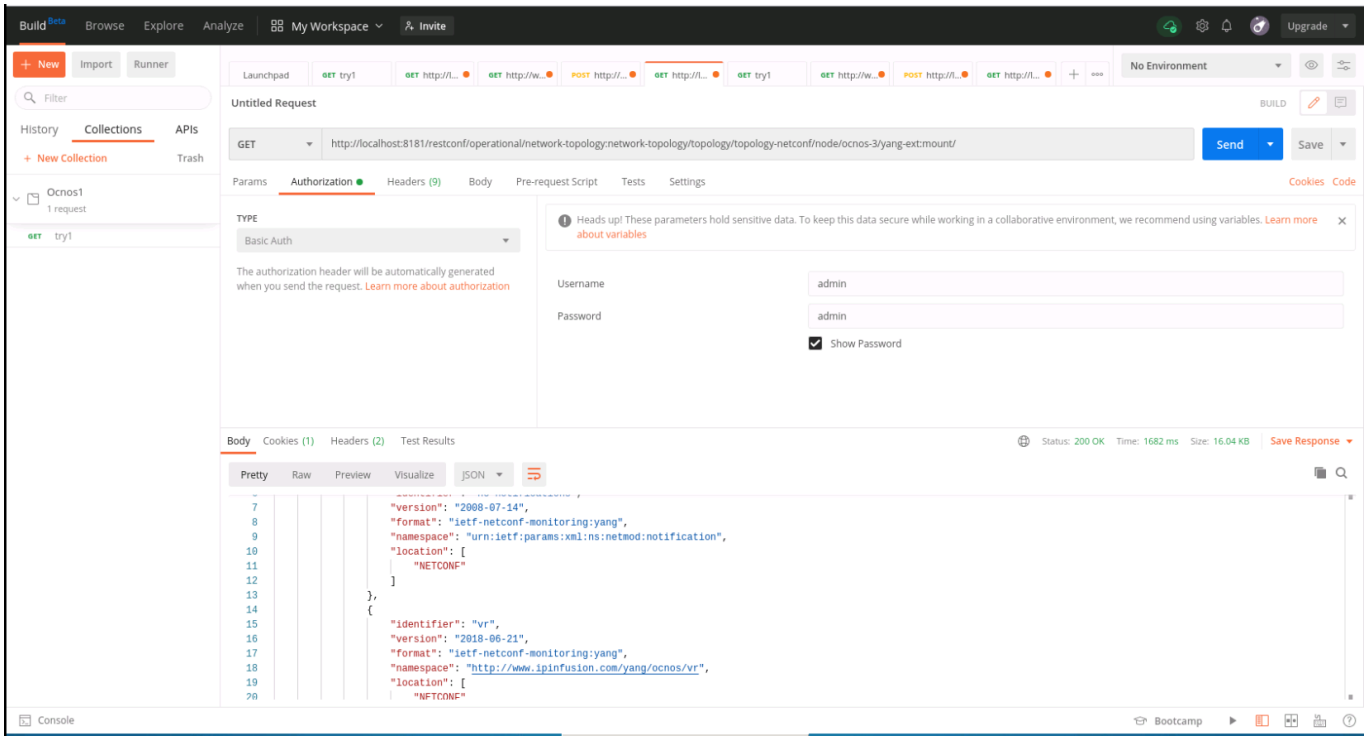
The POSTMAN desktop version should be running in the system while the web version is accessed at this link:

https://web.postman.co/

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

All the RESTCONF operations can be performed using POSTMAN tool.

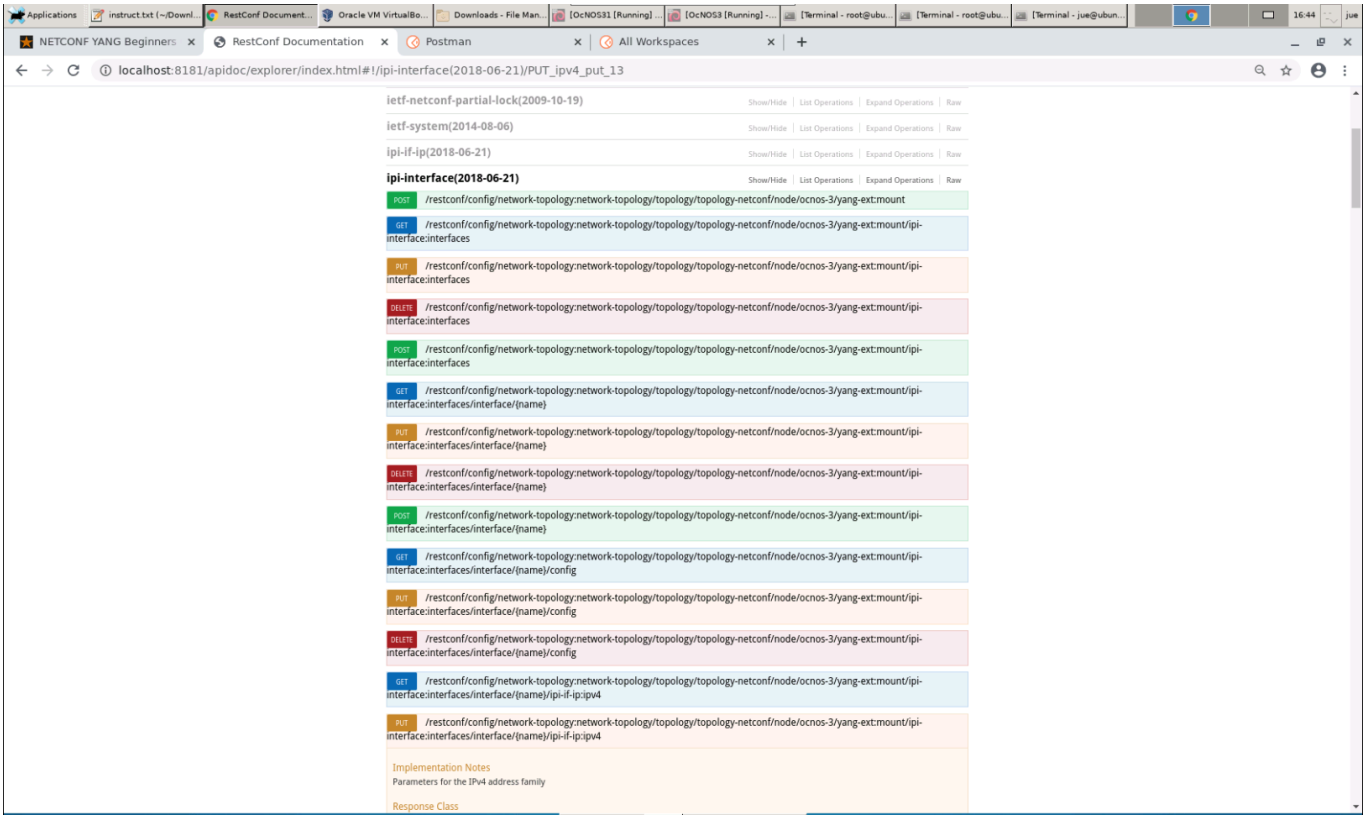
Configuration using POSTMAN is shown later in the section "edit-config".




# Edit-config

## Edit-config using API-DOCS

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




## Edit-config using Terminal

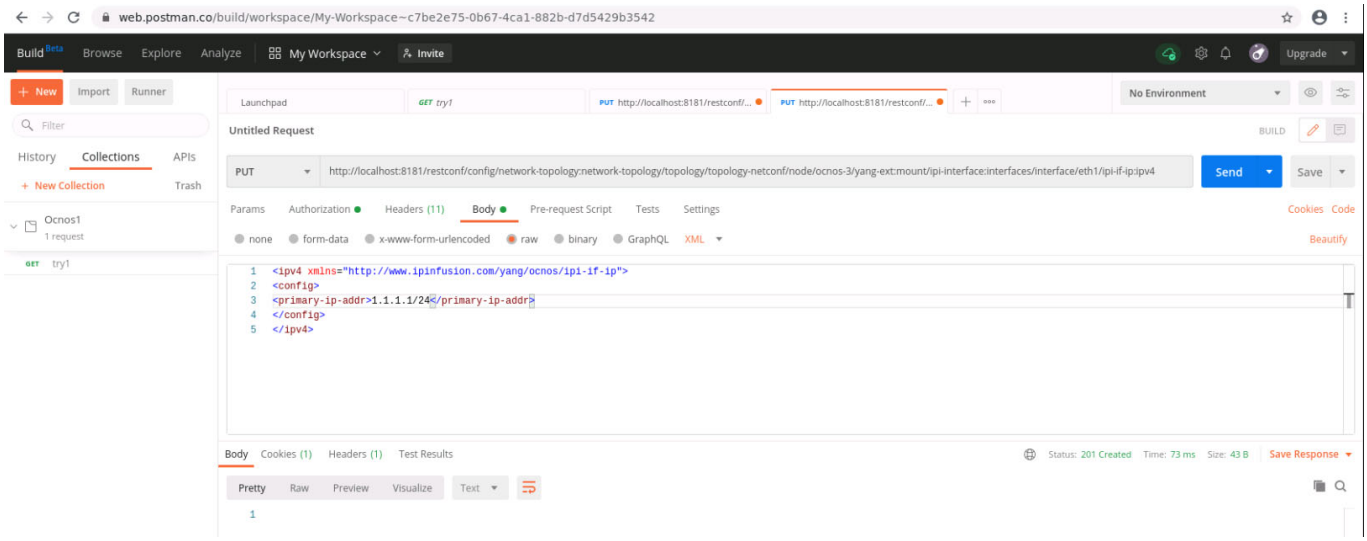
```
$ curl -user "admin":"admin" -H "Content-type: application/xml" -X PUT http://
localhost:8181/restconf/config/network-topology:network-topology/topology/topology-
netconf/node/ocnos3/yang-ext:mount/ipi-interface:interfaces/interface/eth1/ipi-if-
ip:ipv4 -d '@interface_config.xml'
```

The interface_config.xml file contents are:

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

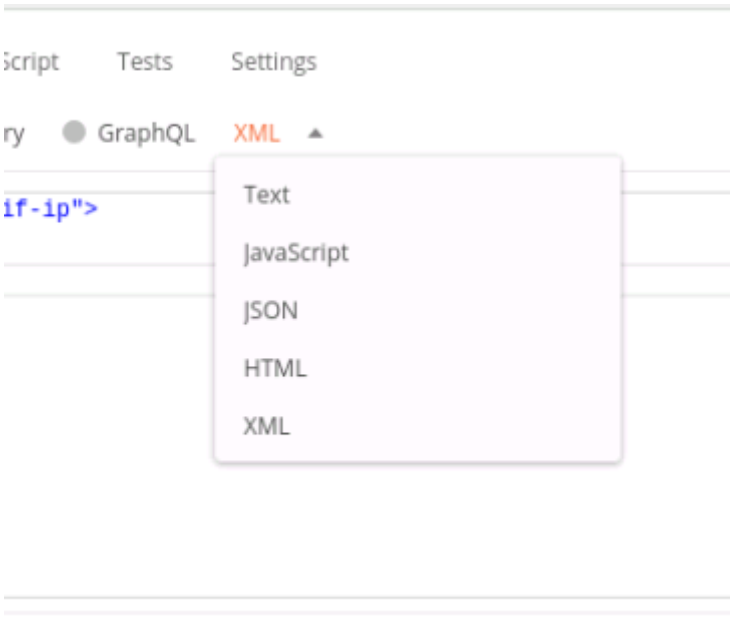
## Edit-config using POSTMAN

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




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

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




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

# Known Issues and Troubleshooting

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

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

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

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

with the provided JAR file.

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

### To apply the PATCH using .JAR files

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

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

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

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

### Unavailable-capabilities in hello message

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

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

`opendaylight-0.12.2/cache/schema/`

and attempt to re-connect with netconf-device.

### JAVA_HOME not set

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

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

To check where the Java binary resides:

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

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

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

Nothing to configure.

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

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

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

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

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

### OcNOS devices with different product versions cannot connect simultaneously

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

Also to connect a device with a different OcNOS product version (when it was connected with other one earlier), 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 they 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`.

TCP port number 6513 is used by the NetConf server to listen for TCP connections established by NetConf over TLS clients.

## 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-----
MIICXDCCAUQCAQAwFzEVMBMGA1UEAwwMMTAuMTIuOTMuMTExMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEA7dTLZpS7sTCrFdP4gQD6X4+PHtYKK7jQdSaM
WaX20dkPqxN8/6VHuY3+fE13eeUSPrDbdSOzPq6f/Rdd/xG76Qi95yYKZnUkTz1/
sF50LfdxL2u+Xg25TA4+Lh6EHYxbmLwX6/o4b8E+78CH+NftH6g1/2YN14PC3lI1
Jlar+YEAzlyy8q1RAHbiazyJmgR0n4KxdtU/c/nzU8zP4OWtJdNquAAMkQUcWwBg
0a88FpFEm5SkQRslLX2BdofpO0SCiMJVkYRJH8yirxKwHx7JBg7EtDovofY585rj
zBZBTycsWYrpWGh+YUm+5ZbEHN+NC2r9UqvLVuHyYWRPB40jxQIDAQABoAAwDQYJ
KoZIhvcNAQELBQADggEBAD0/4MxtfJrz3jeBAMUwIjTBFauTI2rJ0AMxjIcPfvel
wLi/nK4HBU3Ucg9yUqfPYwfi3gLa901AJT5UVPZV0C783nNBrR9GRa91rL0+0Ksa
```

```
hptlTCvFug/N7oEIADQ2IHskNTyCSW7MeJaz06amhiGHp+QNjNulAmsXUjPOzKsJ
OILOmBvD10N+GvtvBMhJrxDKzCda9auSdk1If1BRdfmNttnfthoK3PWK5kmkyhlr
l4mz3Mvd9E5UvsMfL8u72FibwaHa3b862/YQdbidN2LS8xjCZTxeitbJooLzywhb
hR99BENUG8zTbmIa8BD2Nt8F3mlo/31kD8eO2QTLqfI=
-----END CERTIFICATE REQUEST-----
PE3#
```

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

# 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. This process, as shown in Figure 15-2, is standardized by IETF in RFC 8071.

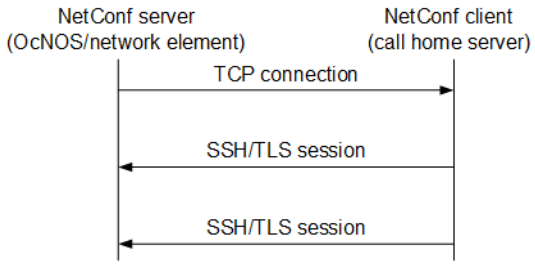



**Figure 15-2: RFC 8071 NetConf call home functionality**

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 both the initial deployment and ongoing management of networking elements.

## 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*.