ORAN-WG4.MP.0-v01.00

Technical Specification

O-RAN Alliance Working Group 4

Management Plane Specification

Revision History

Date	Revision	Author	Description
2019.03.11	01.00	O-RAN-WG4	First published version based on import of xRAN M-Plane

Contents

Chapter 1 Introductory Material

1.1 Scope

This Technical Specification has been produced by the O-RAN.org.

The contents of the present document are subject to continuing work within O-RAN WG4 and may change following formal O-RAN approval. Should the O-RAN.org modify the contents of the present document, it will be re-released by O-RAN Alliance with an identifying change of release date and an increase in version number as follows:

Release x.y.z

where:

x the first digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc. (the initial approved document will have x=01).

y the second digit is incremented when editorial only changes have been incorporated in the document.

z the third digit included only in working versions of the document indicating incremental changes during the editing process.

The present document specifies the management plane protocols used over the fronthaul interface linking the O-RU (O-RAN Radio Unit) with other management plane entities, that may include the O-DU (O-RAN Distributed Unit) as well as other Network Management Systems (NMS).

1.2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document in Release 15.

[1] 3GPP TR 21.905: “Vocabulary for 3GPP Specifications”

[2] ORAN-WG4.CUS.0-v01 “Control, User and Synchronization Plane Specification”, O-RAN Alliance, Working Group 4

[3] RFC 6241, “Network Configuration Protocol (NETCONF)”, IETF, June 2011

[4] RFC 7950, “The YANG 1.1 Data Modeling Language”, IETF, August 2016

[5] RFC 6242, “Using the NETCONF Protocol over Secure Shell (SSH)”, IETF, June 2011

[6] RFC 4252, “The Secure Shell (SSH) Authentication Protocol”, IETF, January 2006

[7] RFC 4253, “The Secure Shell (SSH) Transport Layer Protocol”, IETF, January 2006

[8] RFC 2132, “DHCP Options and BOOTP Vendor Extensions”, IETF, March 1997

[9] RFC 3925, “Vendor-Identifying Vendor Options for Dynamic Host Configuration Protocol version 4 (DHCPv4)”, IETF, October 2004

[10] RFC 2131, “Dynamic Host Configuration Protocol”, IETF, March 1997

[11] RFC 4862, “IPv6 Stateless Address Autoconfiguration”, IETF, September 2007

[12] RFC 3315, “Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”, IETF, July 2003

[13] RFC 3736, “Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6”, IETF, April 2004

[14] draft-ietf-netconf-zerotouch, “Zero Touch Provisioning for NETCONF or RESTCONF based Management”, IETF, work in progress

[15] RFC 8071, “NETCONF Call Home and RESTCONF Call Home”, IETF, February 2017

[16]	SFF-8472v11, "Diagnostic Monitoring Interface for Optical Transceivers", SFF Committee, September 2010
[17]	IEEE 802.1ag, "IEEE Standard for Local and Metropolitan Area Networks Virtual Bridged Local Area Networks Amendment 5: Connectivity Fault Management", IEEE, 2007
[18]	RFC 862, "Echo Protocol", IETF, May 1983
[19]	MEF.38, "Service OAM Fault management YANG Modules", Metro Ethernet Forum, April 2012
[20]	RFC 7895, "YANG Model Library", IETF, June 2016
[21]	RFC 5277, "NETCONF Event Notifications", IETF, July 2008
[22]	G.8275.1, "Precision time protocol telecom profile for phase/time synchronization with full timing support from the network", ITU, June 2016
[23]	G.810, "Definitions and terminology for synchronization networks", ITU, August 1996
[24]	1588v2-2008, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE, 2008
[25]	Y.1731, "Operation, administration and maintenance (OAM) functions and mechanisms for Ethernet based networks", ITU, August 2015
[26]	AISG 2.0, "Control interface for antenna line devices", Antenna Interface Standards Group, June 2006
[27]	3GPP 25.462, "UTRAN Iuant interface: Signalling transport", 3GPP
[28]	3GPP 25.466, "UTRAN Iuant interface: Application part", 3GPP
[29]	3GPP 25.463, "UTRAN Iuant interface: Remote Electrical Tilting (RET) antennas Application Part (RETAP) signalling", 3GPP
[30]	ITU X.733, "Information Technology – Open Systems Interconnection - System Management: Alarm Reporting Function", 1992
[31]	RFC 6187, "X.509v3 Certificates for Secure Shell Authentication", IETF, March 2011
[32]	3GPP TS 36.213, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures", 3GPP, V13.6.0 (2017-06)

1.3 Definitions and Abbreviations

1.3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

Antenna Line: connection between O-RU and antenna

C-Plane: Control Plane: refers specifically to real-time control between O-DU and O-RU, and should not be confused with the UE's control plane

c_eAxC: component eAxC: a portion of an eAxC flow assigned to a specific O-DU processing element.

DL: DownLink: data flow towards the radiating antenna (generally on the LLS interface)

eAxC: extended Antenna-Carrier: a data flow for a single antenna (or spatial stream) for a single carrier in a single sector.

LLS: Lower Layer Split: logical interface between O-DU and O-RU when using a lower layer (intra-PHY based) functional split.

LLS-U: Lower Layer Split User-plane: logical interface between O-DU and O-RU when using a lower layer functional split.

LLS-C: Lower Layer Split Control-plane: logical interface between O-DU and O-RU when using a lower layer functional split.

LLS-S: Lower Layer Split Synchronization-plane: logical interface between O-DU and O-RU when using a lower layer functional split.

High-PHY: those portions of the PHY processing on the O-DU side of the fronthaul interface, including FEC encode/decode, scrambling, and modulation/demodulation.

Low-PHY: those portions of the PHY processing on the O-RU side of the fronthaul interface, including FFT/iFFT, digital beamforming, and PRACH extraction and filtering.

M-Plane: Management Plane: refers to non-real-time management operations between the O-DU and the O-RU

NMS: A Network Management System dedicated to O-RU operations

Port: End of a transport link – in most cases this is an optical port

Port Number: A number which identifies a port (see Port). In case of SFP/SFP+ port, port number value is 0 to N-1 where N is number of ports in the device. Numbers 0 to N-1 are assigned to ports in order following order of labels on the device (labels for ports are not necessarily numbers starting from zero)

O-DU: O-RAN Distributed Unit: a logical node hosting PDCP/RLC/MAC/High-PHY layers based on a lower layer functional split.

O-RU: O-RAN Radio Unit: a logical node hosting Low-PHY layer and RF processing based on a lower layer functional split. This is similar to 3GPP's "TRP" or "RRH" but more specific in including the Low-PHY layer (FFT/iFFT, PRACH extraction).

O-RU Controller: A network function that is permitted to control the configuration of an O-RU. Examples of O-RU controllers include, an O-DU, a classical NMS, or other network automation platforms.

S-Plane: Synchronization Plane: refers to traffic between the O-RU or O-DU to a synchronization controller which is generally an IEEE-1588 Grand Master (however, Grand Master functionality may be embedded in the O-DU).

Spatial stream: the data flow on the DL associated with precoded data (may be same as layers or different if there is expansion in the precoding), and on UL associated with the number of outputs from the digital beamforming (sometimes called "beams").

SSM: Synchronization Status Message: part of ITU G.781 and G.8264 standards.

TRX: Refers to the specific processing chain in an O-RU associated with D/A or A/D converters. Due to digital beamforming the number of TRXs may exceed the number of spatial streams, and due to analog beamforming the number of TRXs may be lower than the number of antenna elements.

U-Plane: User Plane: refers to IQ sample data transferred between O-DU and O-RU

UL: UpLink: data flow away from the radiating antenna (generally on the LLS interface)

Virtual Connection: a connection between O-RU and O-RU controller. This connection is established by means of autodetection procedure and is supervised by supervision procedure.

1.3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

ALD	Antenna Line Device
AVP	Average Power
BCN	BTS Clock Number
CRC	Cyclic Redundancy Check
CUS	Control/User/Synchronization
DHCP	Dynamic Host Configuration Protocol
DMTC	DRS Measurement Timing Configuration
DRS	Discovery Reference Signal
DSCP	Differentiated Services Code Point
HDLC	High-Level Data Link Control
lls-M	Lower Layer Split Management plane

LAA	Licensed Assisted Access
LBM	Loop-Back Message
LBR	Loop Back Reply
LBT	Listen Before Talk
ME	Maintenance Entity
MEP	Maintenance association End Point
NAT	Network Address Translation
NETCONF	Network Configuration Protocol
O-DU	O-RAN Distributed Unit (see definitions section)
O-RU	O-RAN Radio Unit
OMA	Optical Modulation Amplitude
PDV	Packet Delay Variation
QoS	Quality of Service
RET	Remote Electrical Tilt
RPC	Remote Procedure Call
SFP	Small Form-factor Pluggable
sFTP	Secure File Transfer Protocol or SSH File Transfer Protocol
SLAAC	Stateless Address Auto Configuration
SSH	Secure Shell
VLAN	Virtual LAN
YANG	Yet Another Next Generation

1.4 Conventions

This management plane specification includes cross references to a set of associated YANG models. Text may reference particular YANG leafs, notifications and remote procedure calls (RPCs). In order to assist in readability, all cross references to YANG defined elements will keep the identical case format as defined in the corresponding YANG model, with the font-weight set to **bold**. This convention applies only to text and not to YANG elements embedded into figures.

If there is any conflict between the YANG models and the accompanying text description in this specification, the definition of the YANG models shall take precedence.

1.5 Topics for Future Specification Versions

This version of the management plane specification is primarily aimed at providing support those mandatory and optional features defined in version 1.0 of the CUS plane specification, published February, 2019.

The following topics are to be considered for future versions of the specification:

1. Beam Id field interpretation for various types of beamforming
2. Redundancy and failover scenario

Extensions to this version of the O-RAN WG4 Management Plane specification together with corrected errors will be included in the future versions of this document.

The revision statement in the YANG models will be used to describe future revisions to the models that are backwards compatible. Backwards incompatible changes will be addressed by incrementing the number used as part of the model name and namespace, effectively creating a new YANG model. The format of the namespace used in all O-RAN YANG models is “urn:o-ran:”<model-name>“:”<model-number>, where the initial <model-number> used in a newly defined YANG model is “1.0”. Where this document makes reference to models, irrespective of their backward compatibity, a generic <model-number> of “x.y” is used to enable reference to all versions of the namespace for a particular <model-name>.

The revision statement in all YANG models includes a reference statement used to cross-reference to the first version of this document where the corresponding description was introduced. For example, the reference in all revision statements for the initial O-RAN models include cross-reference to “ORAN-WG4.MP.0-v01.00”.

The revision statement of the YANG models also includes a description which is used to track the versioning of the YANG model. All revision statement descriptions will begin with “version ”<a>“.”<b>“.”<c>, where <a>, <b> and <c> are used to reflect the version of the YANG model, where

<a>	corresponds to the first digit of the O-RAN WG4 management plane specification version where the corresponding description was first introduced, corresponding to <x> in sub-section 1.1
<b>	is incremented when errors in the YANG model have been corrected
<c>	is incremented only in working versions of the the YANG model indicating incremental changes during the editing process

Chapter 2 High Level Description

2.1 Top level functional description, terminology, including hybrid, hierarchical

2.1.1 Architecture for O-RAN WG4 Fronthaul functional split

This O-RAN FH specification addresses the lower layer functional split as depicted in Figure 1. Refer to the O-RAN CUS plane specification [2] for more details on the split architecture. The Lower-Layer Split M-plane (LLS-M) facilitates the initialization, configuration and management of the O-RU to support the stated functional split.

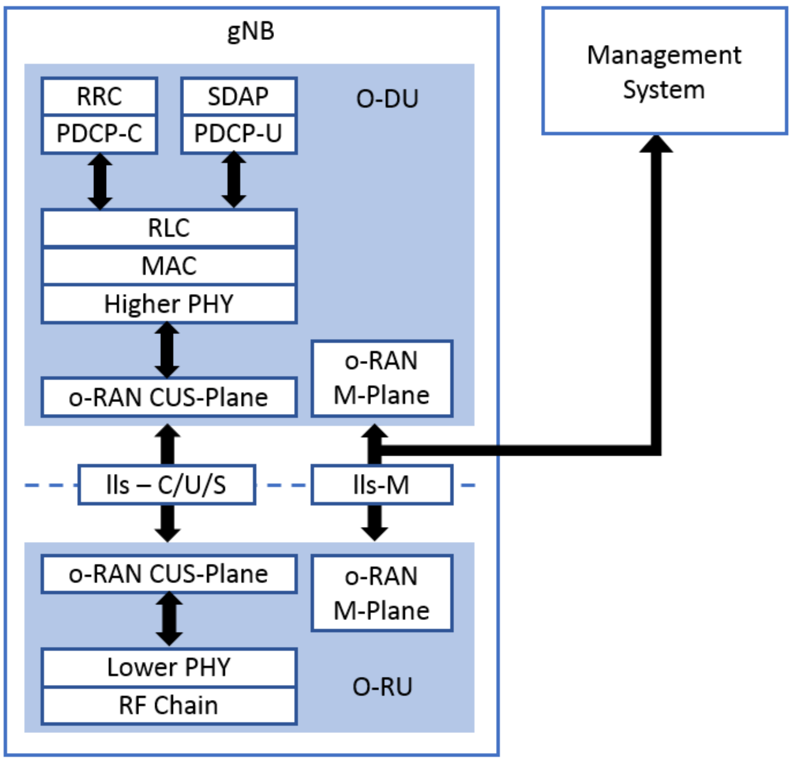


Figure 1 – O-RAN WG4 FH functional split

2.1.2 M-Plane architecture model

A NETCONF/YANG based M-Plane is used for supporting the management features including "start up" installation, software management, configuration management, performance management, fault management and file management towards the O-RU. The M-Plane supports two architectural models:

1. **Hierarchical model**. As shown on the left side Figure 1, the O-RU is managed entirely by one or more O-DU(s) using a NETCONF based M-Plane interface. When the O-RU is managed by multiple O-DUs, it is typically for enabling O-DU and/or transport connectivity redundancy capabilities. Refer to Chapter 3 for more details.

2. **Hybrid model**. As shown on the right side of Figure 1, the hybrid architecture enables one or more direct logical interface(s) between management system(s) and O-RU in addition to a logical interface between O-DU and the O-RU. It should be noted that the NETCONF clients connecting to the O-RU may be of different classes (e.g. O-DU and NMS). For example, functions like O-RU software management, performance management, configuration management and fault management can be managed directly by the management system(s).

In the hybrid model, the O-RU has end to end IP layer connectivity with the NMS. From a physical network point of view, this connectivity could be via the O-DU, where the O-DU is acting as an IP/Ethernet packet forwarder, forwards the packets between O-RU and the NMS. Direct logical communication between an O-RU and NMS can be enabled via O-RUs being assigned routable IPs or local private IPs resolved by a NAT function in the network (or implemented at the O-DU). Refer to chapter 3 for details how O-RU acquires the IP address of O-DU and NMS for the M-plane communication.

As described in Chapter 3, there is no explicit signaling to indicate that an O-RU is operating in a hierarchical or hybrid configuration. All NETCONF servers supporting this M-Plane specification shall support multiple NETCONF sessions, and hence all complaint O-RUs shall be able to support both hierarchical and hybrid deployment.

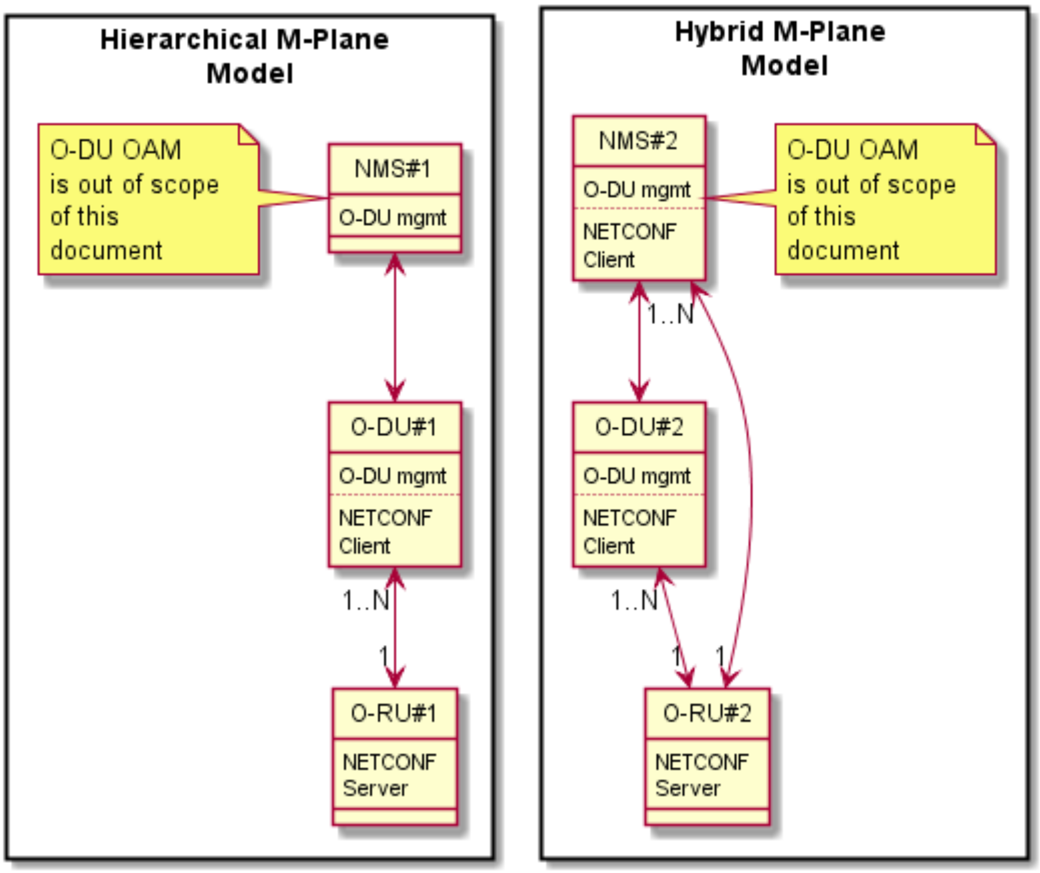


Figure 2 - M-Plane Architecture

NETCONF/YANG is used as the network element management protocol [3] and data modeling language [4]. Use of such a standardized framework and common modeling language simplifies integration between O-DU and O-RU as well as operator network integration (in terms of running service) in case of elements sharing a common set of capabilities. The framework supports integration of products with differing capabilities enabled by well-defined published data models. NETCONF also natively supports a hybrid architecture which enables multiple clients to subscribe and receive information originating at the NETCONF server in the O-RU.

2.1.3 Transport Network

Based on the transport topology, various modes of network connectivity are possible between O-RU and O-DU and NMS.

The basic requirement for M-Plane is to have end to end IP connectivity between the O-RU and the elements managing it (O-DU, NMS, or so called “O-RU Controllers”). The connectivity between the O-DU and NMS and its management plane are not in scope of this specification. IPv4 shall be supported as a mandatory transport protocol for M-Plane and IPv6 support is optional. The M-Plane shall be NAT agnostic for the O-RU to O-DU/NMS connectivity, in order to support a variety of possible network and IP addressing deployments.

2.1.4 M-Plane functional description

The M-Plane provides the following major functionalities to the O-RU. These features are implemented using the NETCONF provided functions.

“Start up” installation

During startup, the O-RU acquires its network layer parameters either via static (pre-configured in the O-RU) or dynamically via DHCP or DHCPv6. During this process it also acquires the IP address of the O-RU controller(s), and establishes the NETCONF connectivity using the “call home” feature. The capability exchange is performed between the client and server as part of the initial NETCONF Hello exchanges. Details of these steps are provided in chapter 3.

SW management

The M-Plane is responsible for software download, installation, validation and activation of new SW when requested by O-RU Controller. The software download is triggered by NETCONF RPC procedures, and the actual software package download is done using sFTP. In this version of the M-Plane specification, sFTP is the only defined file transfer protocol for software and file management.

Configuration management

Configuration management covers various scenarios like Retrieve Resource State, Modify Resource State, Modify Parameters and Retrieve Parameters. NETCONF **get-config** and **edit-config** RPCs shall be used for configuration parameter retrieval and updates at the O-RU

Performance management

Performance management describes the measurements and counters used to collect data related to O-RU operations. The purpose of Performance Management is optimizing the operation of the O-RU.

The measurement results are reported by two options:

1. **YANG Notification**: This option uses the stats definition of YANG model per measurement group. In this case, **get** rpc and/or notification will be used (see Chapter 7 for more details).
2. **File Upload:** This option uses the file upload procedure defined in File management. The measurement results are saved to a data file periodically.

Fault Management

Fault management is responsible for sending alarm notifications to the NETCONF Client. Fault Management allows alarm notifications to be disabled or enabled as well as alarm subscription.

File Management

File management allows the O-RU Controller to trigger an O-RU to perform upload of files stored on O-RU to O-RU Controller. The O-RU may provide different kinds of files and retrieved files can be used for various purposes. Simultaneous multiple file upload operations can be supported under the same sFTP connection between O-RU to O-DU/NMS.

2.2 Interfaces

The M-Plane interface is defined between the O-DU/NMS and the O-RU. The protocol stack of the M-Plane interface is shown in Figure 3. The transport network layer is built on IP transport and TCP/SSH is used to carry the M-Plane message between the DU/NMS and the O-RU.

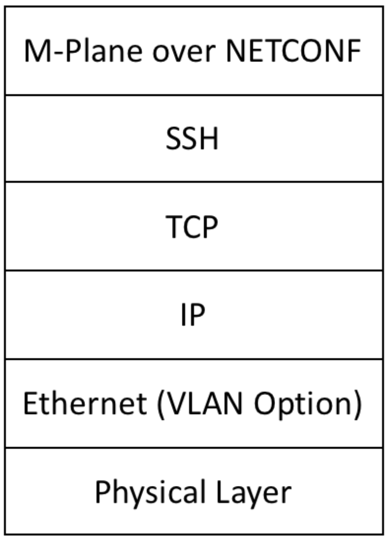


Figure 3 M-plane protocol stack

2.3 YANG Module Introduction

The data models representing the M-Plane are organized as a set of reusable YANG modules. It is also the intent to reuse the publicly available and generic YANG models as much as possible instead of developing customized O-RAN specific modules. Refer to the various chapters, Annex D and the repository of YANG models for more details on each of these modules.

2.4 Security

The M-Plane provides end to end security as a mandatory feature. The security is provided using the SSHv2 layer in accordance with RFC 6242 [5]. RFC 6242 provides the procedure for interoperability with NETCONF implementations. If there are multiple NETCONF sessions established with a single O-RU, each session should be established over a separate SSH tunnel.

Plane	Integrity (protection from modifications)	Confidentiality (encryption protection)	Authentication (validity of the originator)	Remarks
M-Plane	Yes	Yes	Yes	Using the SSHv2 layer used for NETCONF transport

Table 1- M-Plane Security

SSHv2 performs server host authentication, key exchange, encryption, and integrity protection. It also derives a unique session ID that may be used by higher-level protocols. The end point authentication should be done as per RFC 4252 [6]. Chapter 3 describes the authentication approach based on username and password as well as based on X.509 certificates.

The SSHv2 transport level security (encryption algorithms, data integrity algorithms) shall be based on RFC4253 [7]. As per this RFC, 3des-cbc shall be the mandatory cyphering protocol, and rest of the ones listed as optional. For data integrity, hmac-sha1 shall be the mandatory algorithm, and rest of the listed algorithms shall be optional. Public key-based host authentication shall be used for authenticating the server (RFC 4253) by the clients, and username/password based client authentication shall be done by the server as part of the SSH session establishment.

As an additional option, both client and server may implement authentication based on X.509 certificates. With this option, RSA 2048 bit shall be supported for the Public Key algorithm, aes-cbc shall be supported for the cyphering algorithm and hmac-sha256 shall be supported for integrity algorithm.

Note: FIPS-186-3 introduces longer keys but as only FIPS/186-2 is referred in RFC4253 and SHA-1 is used, then key length defined in FIPS-186-2 shall be supported.

Chapter 3 “Start up” installation

This chapter provides the overall start-up mechanism from the power-on of O-RU to available in service.

Pre-condition:

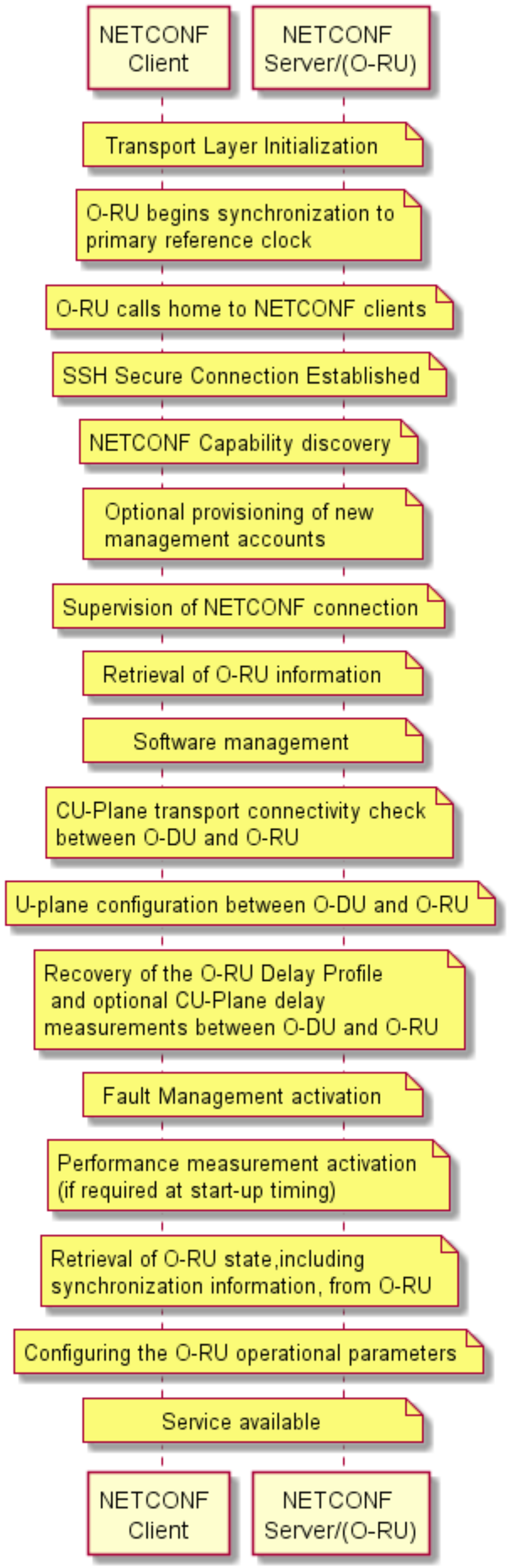
- Power-ON for O-RU/NETCONF Server or O-RU restart operation.

  Power-ON for O-RU controller/NETCONF Client(s).

- Physical interface(s) is(are) connected.

Post-condition:

- O-RU is ready for the radio transmission to the air on at least one carrier if packet transmission received from O-DU
- O-RU is ready for the packet transmission to the O-DU if radio reception received at the air on at least one carrier.
- At least one O-RU Controller/NETCONF client with “super-user” access privileges can control any operation to the O-RU/NETCONF server in O-RU.


NETCONF Client
NETCONF Server/(O-RU)
Transport Layer Initialization
O-RU begins synchronization to primary reference clock
O-RU calls home to NETCONF clients
SSH Secure Connection Established
NETCONF Capability discovery
Optional provisioning of new management accounts
Supervision of NETCONF connection
Retrieval of O-RU information
Software management
CU-Plane transport connectivity check between O-DU and O-RU
U-plane configuration between O-DU and O-RU
Recovery of the O-RU Delay Profile and optional CU-Plane delay measurements between O-DU and O-RU
Fault Management activation
Performance measurement activation (if required at start-up timing)
Retrieval of O-RU state,including synchronization information, from O-RU
Configuring the O-RU operational parameters
Service available
NETCONF Client
NETCONF Server/(O-RU)

1

Figure 4: Overall of Start-Up Installation

At the power-on of O-RU or following an O-RU restart, the following procedures are performed.

1. O-RU performs transport layer resolution (DHCP, MAC, VLAN, IP, etc.) and recovers IP address(es) of O-RU controller(s).
2. O-RU begins synchronization of the O-RU against a Primary Reference Clock. (Note: step 2 may be in parallel with step 1 for some O-RU implementation.)
3. O-RU performs NETCONF Call Home to O-RU controller(s).
4. O-RU controller performs SSH connection establishment.
5. O-RU and O-RU controller perform NETCONF capability discovery.
6. O-RU controller performs optional provisioning of new management accounts (typically only performed once during pre-staging)
7. O-RU and O-RU controller perform supervision of NETCONF connection.
8. O-RU controller performs retrieval of O-RU information.
9. O-RU controller performs SW management.
10. O-RU controller recovers the O-RU delay profile from the O-RU and configuration of C/U-plane transport connectivity between O-DU and O-RU.
11. The O-RU controller configures transport connectivity checking on the O-RU, enabling the O-DU to perform C/U-Plane transport connectivity checking between O-DU and O-RU.
12. O-RU controller performs Fault Management activation.
13. O-RU controller activates performance measurement (if required at start-up timing).
14. O-RU controller retrieves O-RU state, including synchronization information, from O-RU.
15. O-RU controller configures the O-RU operational parameters.
16. Service available.

Note, the synchronization procedures started in step 2 needs to be completed before service is available.

This chapter mainly covers 1, 3 to 7 and 11 as sub-sections.

Cross Reference of other chapters:

The detail of 2. Synchronization management is described in Chapter 10.

The method of 8 and 14 retrieval of O-RU information is described in Chapter 6.

The detail of 9. SW management is described in Chapter 5.

The detail of 10. U-plane configuration is described in Chapter 12.

The detail of 13. Performance management is described in Chapter 7.

The detail of 12. Fault management is described in Chapter 8.

The method of 15. Control to make service available is described in Chapter 12.

3.1 Management Plane Transport aspects

This sub-section provides the M-plane transport establishment scenario between O-RU and O-RU controller(s), such as O-DU and/or NMS. The transport layer address of M-plane is only the target in this section. Transport aspects of the C plane and U plane are covered in Chapter 4.

Pre-condition:

- Physical interface is connected.

- The NETCONF server and NETCONF Client(s) have an identical NETCONF call home port configured, to ensure the NETCONF client listens on the same port used by the NETCONF Server.

Post-condition:

- Transport Layer address(es) for M-plane are known to O-RU and controllers.

- O-RU is aware of the physical port(s) for M-plane, e.g., if there are multiple ports in the O-RU.

- O-RU is aware of the VLAN(s) to be used for M-Plane, e.g., if VLANs are used in the transport network.

- Then O-RU is ready to establish TCP connection for NETCONF call home.

For the transport establishment, there are the following alternatives:

a) Manual transport layer address configuration in O-RU. This configuration contains the addresses for O-RU and NETCONF client(s). The method to manually configure the O-RU is out of scope in this specification. Assuming manual configuration is successful, the NETCONF server shall be able to recover this configured information and use the o-ran-mplane-int.yang model to communicate this operational-state to a NETCONF client.

b) DHCP server provides O-RU's transport layer address information together with the identity of the NETCONF client. This identity encodes either the transport layer address or FQDN of the NETCONF client. If an FQDN is signaled, the O-RU shall use the DNS server address provided by the DHCP server to recover the IP address corresponding to FQDN of the NETCONF client.

c) If IPv6 is supported, Stateless Address Auto-Configuration (SLAAC) is used to configure the O-RU's transport address with the DHCP server providing the identity of the NETCONF client. This identity encodes either the transport layer address or FQDN of the NETCONF client. If an FQDN is signaled, the O-RU shall use the DNS server address provided by the DHCP server to recover the IP address corresponding to FQDN of the NETCONF client.

Note, a NETCONF client can determine whether an O-RU supports IPv6 by using the **get** rpc to recover the list of **interfaces** supported by the O-RU and using the presence of the augmented **ipv6** container in the o-ran-interfaces module to indicate IPv6 is supported.

The O-RU uses the o-ran-dhcp.yang model to be able to expose information signaled by the DHCP server.

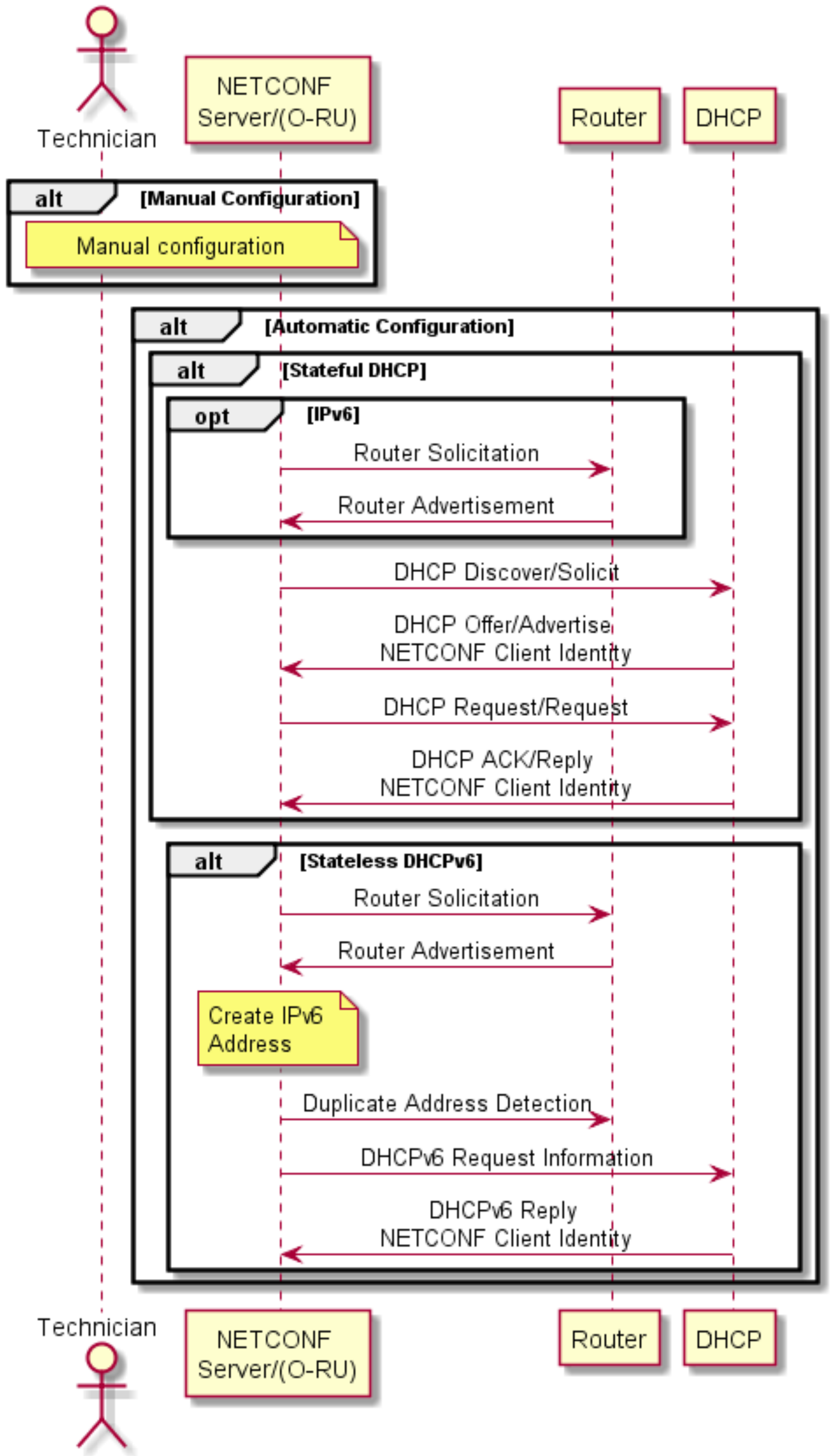


Figure 5: Transport Layer Establishment for M-plane

Transport Layer interface related information for M-plane contains at least the physical port number, the hardware address of the Ethernet port, VLAN-ID, local IP address, remote IP address, Default Gateway address and Subnet mask.

In the case of option b) and c), the following subsections are used:

- O-RU identification in DHCP messages from O-RU.
- VLAN discovery aspect for M-plane.
- IP address assignment to O-RU.
- Discovery of address information of O-RU controller(s).

3.1.1 O-RU identification in DHCP

The O-RU shall identify itself to DHCP servers by using DHCP option(s) using the **vendor-class-data** string within the o-ran-dhcp YANG model. For DHCPv4, there are two alternatives. One uses option 60 Vendor Class Identifier, RFC2132 [8]. The other uses option 124 Vendor Identifying Vendor Class Option, RFC3925 [9]. The O-RU shall support at least one of these options. If the O-RU supports IPv6, then it shall identify itself using the DHCPv6 Vendor Class Option.

DHCPv4 Vendor Class Option:

- Option: 60
- Vendor Class Identifier Option 60: string of the format ““o-ran-ru/<vendor>”, e.g., “o-ran-ru/vendorA

DHCPv4 Vendor-Identifying Vendor Class Option:

- Option: 124
- Enterprise number: O-RAN-alliance 53148
- Vendor-Class-Data: string of the format “o-ran-ru/<vendor>”, e.g., “o-ran-ru/vendorA”

DHCPv6 Vendor Class Option:

- Option: 16
- Enterprise number: O-RAN-alliance 53148
- Vendor-Class-Data: string of the format “o-ran-ru/<vendor>”, e.g., “o-ran-ru/vendorA”

The DHCP Server may use this information when allocating IP addresses or when configuring management plane O-RU Controller(s) information in the O-RU.

3.1.2 Management Plane VLAN Discovery Aspects

The O-RU will be connected to one or more Ethernet ports. The transport systems may be realized such that these Ethernet ports may be configured either as an access port, where untagged Ethernet frames are used, or as a trunk port, where multiple VLANs are configured. At start up, the O-RU will typically not be able to immediately determine whether its ports are attached to remote transport equipment configured for access or trunk mode operation.

Once an O-RU completes its boot-up sequence and Ethernet connectivity is detected on at least one of its Ethernet interfaces, the O-RU starts management plane connection establishment.

The O-RU needs to determine whether it is connected to an access port or a trunk port. In particular, when connected to a trunk port, the O-RU needs to additionally determine the VLAN identity/ies used to support the management plane communication(s). The VLAN(s) used to support management plane communications can be identified by the DHCP server replying to the DHCP DISCOVER message, as described in section 3.1.4.

Note, an O-RU which supports IPv6 may infer that a VLAN is not used to support management plane communications if it receives an IPv6 Router Advertisement without either the “managed address configuration” or “other configuration” bits set.

An O-RU may have been previously configured with management plane VLAN information, for example storing the last VLAN(s) used for management plane connectivity, and/or being previously configured with a range of management plane VLANs by a NETCONF client that has been stored in reset-persistent memory. The O-RU may use this information to optimize its discovery of the VLAN ID(s) used for management plane connectivity.

If the O-RU does not have previously configured management plane VLAN information, the O-RU shall attempt to discover DHCP servers on all of its Ethernet ports using untagged Ethernet frames.

If the O-RU does not receive a DHCP OFFER from a DHCP server using untagged frames, or previously configured VLANs, the O-RU should attempt to contact a DHCP server using individual VLANs on all of its Ethernet ports.

3.1.3 O-RU Management Plane IP Address Assignment

Automatic IP address assignment for the O-RU management plane can be achieved using different techniques:

1. IPv4 configuration using DHCPv4, RFC2131 [10] enables DHCP servers to configure IPv4 network address(es) on the O-RU.

For O-RUs that support IPv6, both stateful and stateless address assignment procedures are supported:

2. IPv6 Stateless Address Auto-Configuration (SLAAC), RFC4862 [11] enables the O-RU to generate link-local and global addresses.

3. IPv6 State-full address configuration uses DHCPv6, RFC3315 [12] and enables DHCP servers to configure IPv6 network address(es) on the O-RU. DHCPv6 is transported using UDP, using the link-local address on the O-RU and a link-scoped multicast address on the DHCP server.

Note: the above does not restrict the realization of the DHCP server, which may be integrated with the O-DU, may be provided by the transport system, or may be accessed via a relay.

The DHCP server should operate using static bindings, i.e., ensuring an O-RU identified by a particular client hardware address will be re-allocated the same management plane IP address, e.g., after performing an O-RU reset procedure.

3.1.4 O-RU Controller Discovery

This section provides how to automatically discover the O-RU Controller address(es).

O-RUs that have obtained their IPv6 addresses by stateless address auto-configuration, shall use stateless DHCPv6, RFC3736 [13], to obtain NETCONF management plane configuration information.

Other O-RUs operating using stateful IPv4 or IPv6 address allocations shall obtain NETCONF management plane configuration information during IP address allocation.

The O-RU shall be able to recover NETCONF client information using the following DHCP Options, draft-ietf-netconf-zerotouch [14]:

- DHCPv4 OPTION V4_ZEROTOUCH_REDIRECT [143]
- DHCPv6 OPTION_V6_ZEROTOUCH_REDIRECT [136]

These options are defined in [14] and are used to deliver bootstrap-server-list information to the O-RU. The O-RU shall use these options to recover the NETCONF client information using the above IANA defined DHCP options. The O-RU is not required to implement the remainder of the zerotouch capabilities defined in [14].

The above DHCP option provided information is encoded as a list of one or more server URIs, of the format “https://<ip-address-or-hostname>[:<port>]” signaled to the O-RU. The DHCP server shall ensure that all NETCONF client information is encoded with these options.

Other O-RUs which have had their IP address(es) manually configured, shall also have their O-RU Controller(s) manually configured.

For IPv4, the O-RU may request the OPTION_V4_ZEROTOUCH_REDIRECT by including its option code in the Parameter Request List (55) in DHCP discover/request messages.

For IPv6, the O-RU may request the OPTION_V6_ZEROTOUCH_REDIRECT option by including the requested option codes in the Option Request Option.

Note, these operations are optional because the DHCP server will already be aware that it is communicating with an O-RU based on the recovered vendor class option.

To enable O-RUs to operate in legacy environments where the DHCP server has not been enhanced with IANA defined DHCP options for zero touch NETCONF capability, an O-RAN defined vendor specific option can be used to signal all NETCONF client information to the O-RU using option 43 for DHCPv4 and option 17 for DHCPv6.

The format of the DHCPv4 option 43 encodes the following information:

Type: [01 – O-RU Controller IP Address] or [02 – O-RU Controller Fully Qualified Domain Name]

Length: Hexadecimal encoding of length of value field in octets

Value:

When Type is 01, the value encodes IPv4 address(es) in hexadecimal format. For example, a single server with IPv4 address 198.185.159.144 will be encoded in an option 43 TLV as

Type 01

Length: 04

Value: C6 B9 9F 90

When Type is 02, this encodes the string representation of domain name, using ACSII encoding (i.e., following for encoding used for the domain name in the Host Name DHCP Option 12). For example, a server with FQDN “controller.operator.com” will be encoded in an option 43 TLV as

Type 02

Length: 17

Value: 63 6F 6E 74 72 6F 6C 6C 65 72 2E 6F 70 65 72 61 74 6F 72 2E 63 6F 6D

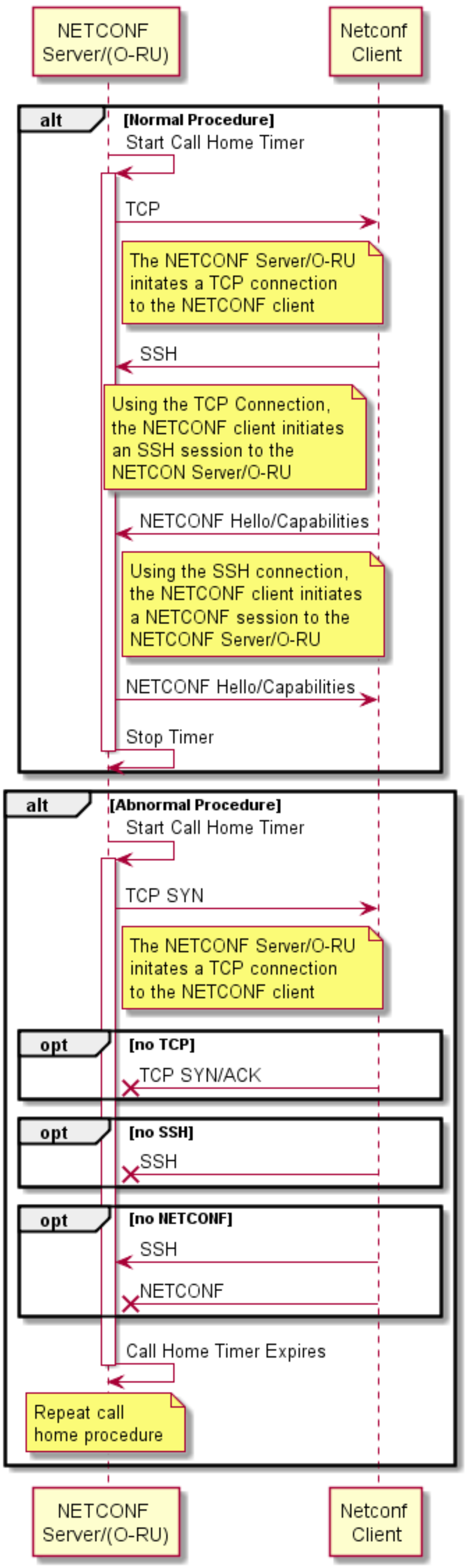
The format of the DHCPv6 option 17 follows the format of the DHCPv4 encoding, with the additional inclusion of an Enterprise Number prior to the TLV option data. The IANA allocated private enterprise number to be used with DHCPv6 option 17 is 53148.

3.2 NETCONF Call Home to O-RU Controller(s)

The O-RU aims to have NETCONF sessions with all of the known O-RU Controller(s), either discovered using the DHCP options defined in Section 3.1.4, provisioned by an existing NETCONF client, or statically configured. An O-RU controller may attempt to autonomously initiate a NETCONF session with the O-RU, e.g., triggered by some out of band interaction with the IP address management functionality. In order to support NETCONF clients corresponding to known O-RU Controllers that either do not attempt to initiate a NETCONF session with the O-RU, or are prohibited from doing so, e.g., because of Network Address Translation limitations, the O-RU shall call home to all known O-RU Controllers with which it does not already have an active NETCONF session.

If the O-RU is unable to establish a NETCONF session with any of the O-RU Controller(s), the O-RU shall use the “re-call-home-no-ssh-timer” to repeatedly re-perform the call home procedure to all with which is does not have an established NETCONF session.

The O-RU shall use RFC 8071 [15] whereby the O-RU (NETCONF Server) initiates a TCP connection to the NETCONF client. The port used by the O-RU shall be the one signaled using the draft-ietf-netconf-zerotouch DHCP option, else, if no port was signaled, the O-RU shall use the IANA-assigned port 4334. When the NETCONF client accepts a TCP connection on the allocated port, it initiates an SSH session with the NETCONF Server. Using this SSH session, the NETCONF client initiates a NETCONF session.


NETCONF Server/(O-RU)
Netconf Client
alt
[Normal Procedure]
Start Call Home Timer
TCP
The NETCONF Server/O-RU initates a TCP connection to the NETCONF client
SSH
Using the TCP Connection, the NETCONF client initiates an SSH session to the NETCON Server/O-RU
NETCONF Hello/Capabilities
Using the SSH connection, the NETCONF client initiates a NETCONF session to the NETCONF Server/O-RU
NETCONF Hello/Capabilities
Stop Timer
alt
[Abnormal Procedure]
Start Call Home Timer
TCP SYN
The NETCONF Server/O-RU initates a TCP connection to the NETCONF client
opt
[no TCP]
TCP SYN/ACK
opt
[no SSH]
SSH
opt
[no NETCONF]
SSH
NETCONF
Call Home Timer Expires
Repeat call home procedure
NETCONF Server/(O-RU)
Netconf Client

1

2 **Figure 6: Outline of NETCONF call home procedure**

The O-RU shall ensure that a persistent connection to any NETCONF client with "sudo" privileges is maintained by actively testing the aliveness of the connection using the keep-alive mechanism defined in [15]. The establishment of NETCONF client privileges is covered in Section 3.4.

3.3 SSH Connection Establishment

The identity of the SSH server (O-RU) shall be verified and authenticated by the SSH client (NETCONF client) according to local policy before password-based authentication data or any configuration or state data is sent to or received from the SSH server.

Public key-based host authentication shall be used for authenticating the server (RFC 4253) by the clients. In addition, server authentication based on X.509 certificates may also be provided [31].

Note: Public key-based host authentication requires that the SSH server (O-RU) public keys are provisioned in the NETCONF client (e.g., O-DU). As an alternative, RFC4251 mentions that "a possible strategy is to only accept a host key without checking the first time a host is connected, save the key in a local database, and compare against that key on all future connections to that host.". This option simplifies the key management procedure as it doesn't require to pre-populate them in O-DU / NMS (SSH client) but obviously at the price of degraded security, therefore the support of this option shall be configurable and left to operator's choice.

3.3.1 NETCONF Security

In this version of O-RU Management Plane Specification, the security of the NETCONF protocol is realized using SSHv2. O-RAN NETCONF implementations shall implement RFC 6242 [5] that addresses earlier delimiter issues when using RFC 4742.

As the default, the NETCONF Server in the O-RU shall provide access to the NETCONF subsystem only when using SSH established using the IANA-assigned TCP port 830 [5]. The O-RU may be configured to additionally allow access to the NETCONF subsystem over other ports.

If multiple NETCONF sessions are established to an O-RU, those sessions shall be established over separate SSH tunnels.

3.3.2 NETCONF Authentication

This version of the O-RU Management Plane Specification uses password authentication method for SSHv2 [6]. In addition, client authentication based on X.509 certificates may also be provided [31].

Upon initial system initialization, the O-RU is configured with a default account. The specific details of the default account are to be agreed between operator and vendor. An example of a default user account is one with username "oranuser" and password "o-ran-password".

Note: As the default account may be operator specific, this may require that the O-RU provides facilities to configure securely this default account at installation time (i.e. before the O-RU is connected to the O-DU).

The default account is member of the "sudo" access control group (see Section 3.4 for details of groups/privileges) as it can be used to create other accounts (see Section 3.3.3).

The identity of the SSH client (NETCONF client) shall be verified and authenticated by the SSH server (O-RU) according to local policy to ensure that the incoming SSH client request is legitimate before any configuration or state data is sent to or received from the SSH client.

Upon initial system initialization, the NETCONF client can authenticate itself to the O-RU using SSH Authentication, with the agreed default username and password.

If authentication based on X.509 certificates according to [31] is supported by SSH client and server, the certificates need to be installed at initial system initialization, or can be obtained through certificate enrolment with operator's PKI (certificate enrolment as defined by 3GPP with CMPv2 protocol between the NE and the operator's CA).

For the purposes of user authentication, the mapping between certificates and user names is provided by the SubjectAltName field of the X.509 certificate, which means that the user name is coded in the subjectAltName. A specific default coding in the SubjectAltName is defined in this document to map to the default account.

3.3.3 User Account Provisioning

The NETCONF client with suitable privileges may provision user accounts on the O-RU, including the accounts (users) name, password, group (see Section 3.4 for details of groups/privileges) and whether a particular account is enabled or disabled.

- The username **name** is a string between 3-32 characters. The first character must be a lowercase letter. The remaining characters can be a lowercase letter or a number.
- The **password** is a string between 8-128 characters. Allowed characters in the password field include lowercase and uppercase letters, numbers and the special characters: ! $ % ^ ( ) _ + ~ { } [ ] . –
- The access control **group** associated with an account (see Section 3.4 for details of groups/privileges).
- Whether an account is **enabled.** The YANG model ensures that at least one user account is always enabled on the O-RU

The new account information (user **name**, **password**, access **group** and whether the account is **enabled**) shall be stored in reset-persistent memory in O-RU.

When certificate based client authentication is used no password needs to be provisioned. At time of SSH connection, user's authorization is done based on the X.509 certificate's SubjectAltName field that codes the associated account's name.

When other user account (sudo) is created, the NETCONF client closes existing NETCONF session as described in section 3.7. Then, the O-RU disables the default account and default account stays disabled over the resets. The default account becomes enabled when the O-RU is reset to the factory default software by following the procedures defined in subsection 5.7. Any other way to enable the default account is not precluded as O-RU vendor implementation matter.

3.4 NETCONF Access Control

This subsection provides the access control for NETCONF client. Its motivation is when multiple NETCONF clients (users) exist, access control mechanism allows to limit some operation for one client but full access for the other client. In particular, for hybrid access configuration as introduced in Chapter 2, this allows the privileges associated with the NETCONF client in the O-DU to be distinct and different from the privileges associated with the NETCONF client in the NMS

In order to support interoperable access control management, the NETCONF Server shall use the IETF NETCONF Access Control Model [RFC8341].

Currently four access control **groups** are defined per SSH session: "sudo", "nms", "fm-pm", and "swm". The table below maps the group **name** to different privileges. Privileges are defined per namespace for read "R", write "W" and execute "X" rpc operations or subscribe to Notifications.

Module Rules	sudo	nms	fm-pm	swm
"urn:o-ran:supervision:x.y"	--X	---	---	---
"urn:o-ran:hardware:x.y"	R-X	R-X	---	---
"urn:ietf:params:xml:ns:yang:ietf-hardware"	RWX	RWX	R-X	---
“urn:ietf:params:xml:ns:yang:iana-hardware”	RWX	RWX	R-X	---
"urn:o-ran:user-mgmt:x.y"	RW- note1	---	---	---
"urn:o-ran:fm: x.y "	R-X	R-X	R-X	---
"urn:o-ran:fan: x.y "	R--	R--	R--	---
"urn:o-ran:sync: x.y "	RW-	RW-	R--	---
"urn:o-ran:delay: x.y "	RW-	R--	R--	---
"urn:o-ran:module-cap: x.y "	R--	R--	R--	---
"urn:o-ran:udpecho: x.y "	RW-	R--	---	---
"urn:o-ran:operations: x.y "	R-X	R--	R--	---
"urn:o-ran:uplane-conf: x.y "	RWX	RWX	R--	---
“urn:o-ran:beamforming: x.y”	RWX	RWX	R--	---
"urn:o-ran:lbm: x.y "	RWX	RW-	R--	---
"urn:o-ran:software-management: x.y "	RWX	RWX	R--	RWX
"urn:o-ran:file-management: x.y "	--X	--X	--X	---
" urn:o-ran:message5: x.y "	RW-	RW-	R--	---
"urn:o-ran:performance-management: x.y "	RWX	RWX	RWX	---
"urn:o-ran:transceiver: x.y "	RW-	RW-	R--	---
"urn:o-ran:externalio: x.y "	RWX	RWX	---	---
"urn:o-ran:ald-port: x.y "	RWX	RWX	---	---
"urn:o-ran:interfaces: x.y "	RWX	RWX	R--	---
"urn:ietf:params:xml:ns:yang:ietf-ip"	RW-	RW-	R--	---
"urn:ietf:params:xml:ns:yang:ietf-interfaces"	RW-	RW-	R--	---
“urn:ietf:params:xml:ns:yang:iana-if-type”	R--	R--	R--	R--
"urn:ietf:params:xml:ns:yang:ietf-inet-types"	R--	R--	R--	R--
"urn:o-ran:processing-elements: x.y "	RW-	RW-	R--	---
" urn:o-ran:mplane-interfaces: x.y "	RW-	RW-	R--	---
"urn:o-ran:dhcp: x.y "	R--	R--	R--	---
"urn:ietf:params:xml:ns:yang:ietf-dhcpv6-types"	RW-	RW-	---	---
“urn:o-ran:ald: x.y”	--X	---	---	---
“urn:o-ran:troubleshooting: x.y”	--X	--X	--X	---
"urn:o-ran:laa: x.y "	RW-	RW-	---	---
"urn:o-ran:laa-operations: x.y "	--X	---	---	---
“urn:ietf:params:xml:ns:yang:ietf-netconf-acm”	RWX	R--	R--	R--
"urn:ietf:params:xml:ns:yang:ietf-yang-library"	RWX	RWX	RWX	RWX
“urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring”	RWX	RWX	RWX	RWX
Note1: The rule list for "urn:o-ran:user-mgmt:1.0” shall additionally deny reading of the password leaf by any NETCONF client				

Table 2: Mapping of account groupings to O-RU module privileges

This mapping shall be encoded in the **rule** list in ietf-netconf-acm.yang model. This rule list shall be unmodifiable by any NETCONF client.

The same model is responsble for configuring the mapping between different **user-names** and **groups**.

3.5 NETCONF capability discovery

The O-RU advertises its NETCONF capabilities in the NETCONF Hello message. The Hello message provides an indication of support for standard features defined in NETCONF RFCs as well as support for specific namespaces.

NETCONF capabilities are exchanged between the O-RU and the NETCONF client(s). Examples of capabilities include [3]:

- Writable-running Capability
- Candidate Configuration Capability and associated Commit operation
- Discard change operation
- Lock and un-lock operations
- Confirmed commit Capability
- Cancel commit operation
- Rollback on error capability
- Validate Capability
- Startup configuration capability
- URL capability
- XPATH capability
- Notifications
- Interleave capability

All O-RAN O-RUs shall support NETCONF Notifications and at least one of the writeable-running and candidate configuration capabilities.

The NETCONF client uses the **get** RPC together with sub-tree based <filter> and XPATH based <filter> to recover particular sub-trees from the O-RU. Please see Chapter 6 for more information on NETCONF based configuration management.

In order to avoid interactions between the operation of supervision watchdog timer (see section 3.6) and the confirmed commit timer (default value set to 600 seconds in RFC 6241), when using the NETCONF confirmed commit capability, a NECTONF client with "sudo" privileges shall ensure the confirmed-timeout is less than the **supervision-notification-interval** timer (default value 60 seconds in o-ran-supervision.yang).

3.6 Monitoring NETCONF connectivity

The O-RU operates a watchdog supervision timer to ensure that it has continued connectivity to at least one NETCONF client which has "sudo" access control privileges, as described in section 3.4. This supervision is intended to be used with the NETCONF client associated with the operation of the peer to the O-RAN Radio's lower layer split. Additionally, the O-RU provides NETCONF Notifications to indicate to remote systems that its management system is operational.

The NETCONF client is responsible for automatically enabling the operation of the supervision watchdog timer by creating supervision-notification subscription, whenever the O-RU has an established NETCONF session to a client with "sudo" privileges. After it has been enabled, the O-RU is responsible for repeatedly sending **supervision-notification**. After that NETCONF client is responsible for sending **supervision-watchdog-reset** RPC and server shall send next notification timestamp in reply. When resetting the watchdog timer, the NETCONF client may configure a new value for the watchdog timer. Client can set new value of watchdog timer without receiving **supervision-notification** from O-RU. New value is taken into use immediately with respect to **supervision-watchdog-reset** RPC content. Next notification should be expected not later than at the moment addressed in timestamp provided by RPC reply.

If the watchdog timer expires, the O-RU will enter "supervision failure" condition, as described in Chapter 11. If all NETCONF sessions to NECTONF clients with "sudo" privileges are closed, the O-RU shall immediately disable operation of the supervision timer.

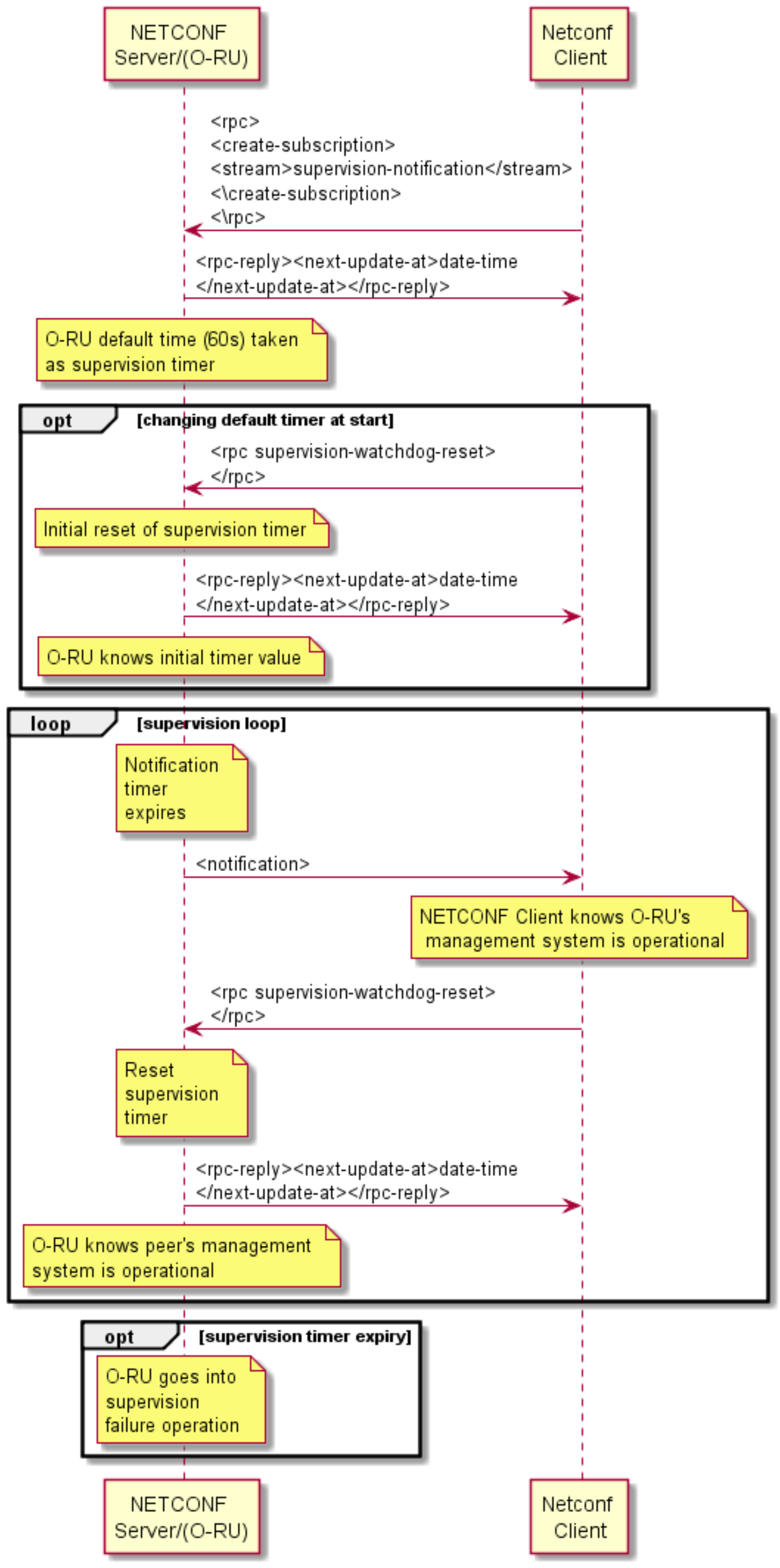

NETCONF Server/(O-RU)
Netconf Client
<rpc>
<create-subscription>
<stream>supervision-notification</stream>
<\create-subscription>
<\rpc>
<rpc-reply><next-update-at>date-time
</next-update-at></rpc-reply>
O-RU default time (60s) taken as supervision timer
opt [changing default timer at start]
<rpc supervision-watchdog-reset>
</rpc>
Initial reset of supervision timer
<rpc-reply><next-update-at>date-time
</next-update-at></rpc-reply>
O-RU knows initial timer value
loop [supervision loop]
Notification timer expires
<notification>
NETCONF Client knows O-RU's management system is operational
<rpc supervision-watchdog-reset>
</rpc>
Reset supervision timer
<rpc-reply><next-update-at>date-time
</next-update-at></rpc-reply>
O-RU knows peer's management system is operational
opt [supervision timer expiry]
O-RU goes into supervision failure operation
NETCONF Server/(O-RU)
Netconf Client

Figure 7: Monitoring NETCONF Connectivity

Note: This figure uses **create-subscription** for the single stream "**supervision-notification**". In order to subscribe multiple notifications, the appropriate **create-subscription** message is required. Please refer to setion 8.2 for the appropriate example of **create-subscription** of multiple notifications.

3.7 Closing a NETCONF Session

A NETCONF client closes an existing NETCONF session by issuing the RPC **close-session** command. The O-RU shall respond and close the SSH session. The O-RU shall then re-commence call home procedures, as described in Section 3.2.

Note, under normal operations, it is expected that at least one NETCONF session with “sudo” privileges is long-lived and used to repeatedly reset the O-RU’s supervision watchdog timer.

Chapter 4 O-RU to O-DU Interface Management

An O-RU has a number of network interfaces, including Ethernet, VLAN and IP interfaces. This section describes the management of these network interfaces.

4.1 O-RU Interfaces

The O-RU’s configuration for its interfaces is defined using the o-ran-interfaces.yang module. This module augments the standard ietf-interfaces.yang and ietf-ip.yang modules. The O-RU’s interfaces are built on a layering principle where each interface has a unique **name**.

All interfaces are referenced by their **port-number** and **name.** The base interface corresponds to the Ethernet interface. These leafs describe the maximum transmission unit (**l2-mtu**), the hardware-address as well as optional alias mac addresseses that may be used to transport the CU plane. Above the Ethernet interface are VLAN interfaces. Both Ethernet and VLAN interfaces can support IP interfaces. IP interfaces are defined using the standard ietf-ip.yang model. Accordingly, each IP interface can have an IPv4 and/or optional IPv6 interface(s) defined. Operational state associated with these interfaces provide additional detail of the layer 3 configuration, including prefix(es), domain name servers and default gateway addresses.

Finally, leafs associated with CoS and DSCP marking are defined, enabling independent configuration of CoS and DSCP markings for u-plane, c-plane and m-plane traffic. As a default, all user-plane flows are marked identically by the O-RU. Optionally, the interfaces can be configured to support enhanced user plane marking for up-link traffic whereby different CoS or DSCP values can be configured. This enables individual receive endpoints in the O-RU to be configured with different markings to then enable differentiated handling of up-link flows by the transport system.

Because the o-ran-interfaces model defines augments to the ietf-interfaces model, the O-RU can leverage the definition of operational state in ietf-interfaces to optionally report packet and byte counts on a per interface basis. A single RPC is defined in the o-ran-interfaces module, to enable these counters to be reset.

4.2 Transceiver

The o-ran-transceiver YANG module is used to define operational state for the pluggable transceiver module (like SFP, SFP+ and SFP28). Each transceiver is associated with a unique **interface-name** and **port-number**.

A digital diagnostic monitoring interface for optical transceivers is used to allow access to device operating parameters. As specified in SFF-8472 [16], data is typically retrieved from the transceiver module in a file. This file may be obtained from O-RU by the NETCONF client. Please see Chapter 9 for more details.

The byte with offset i (i=0, …, 511) from the beginning of the file is the byte read from data address i of the transceiver memory at two-wire interface address 0xA0 if i<256, otherwise it is the byte read from data address i-256 of the transceiver memory at two-wire interface address 0xA2. The retrieved data is stored in the file without any conversion in binary format.

The O-RU stores data from the transceiver module on transceiver module detection during startup. The data from the transceiver module is saved in the file. A NETCONF client can upload it by using the File Upload procedure defined in chapter 9. The O-RU does not synchronize contents of the file with transceiver memory in runtime, therefore bytes representing dynamic information are expected to be outdated. The O-RU does not remove the file on transceiver module removal. If a transceiver module is inserted during File Upload procedure then the procedure may provide a file with previous content or fail (with failure reason as listed in File Upload procedure). If the O-RU is unable to retrieve the data from the transceiver module or it is not present then the O-RU does not create the file or removes the file created earlier (note that File Upload procedure requesting non-existing file shall fail).

The file name shall have the following syntax:

sfp_{portNumber}.sffcap

where {portNumber} is the value of **port-number** leaf of the corresponding list of port-transceiver data. Examples: sfp_0.sffcap, sfp_1.sffcap.

4.3 C/U Plane VLAN Configuration

Within the o-ran-interfaces YANG model, each named Ethernet interface includes a leaf to indicate whether VLAN tagging is supported. By default, VLAN tagging is enabled on all interfaces. This permits an O-RU to autonomously discover that it is connected to a trunk port, as described in Section 3.1.2.

When an O-RU is connected to a trunk port, VLANs will also typically be assigned to the C/U plane connections. The VLAN(s) used to support C/U plane transport may be different from the VLAN(s) used to support management plane connectivity. When different VLANs are used, the C/U plane VLANs shall be configured in the O-RU by the NETCONF client. In such circumstances, as defined in o-ran-interfaces, the NETCONF client shall configure separate named interfaces for each active VLAN. This configuration will define a C/U-Plane named VLAN interface as being the **higher-layer-if** reference for the underlying Ethernet interface and the underlying Ethernet interface is defined as being the **lower-layer-if** reference for the named VLAN interface.

4.4 O-RU C/U Plane IP Address Assignment

In this release, the support for C/U plane transport over UDP/IP is optional and hence this section only applies to those O-RUs that support this optional capability.

An O-RU that supports C/U plane transport over UDP/IP shall support IPv4 and optionally IPv6 based transport. A NETCONF client can determine whether an O-RU supports IPv6 by using the **get** rpc to recover the list of **interfaces** supported by the O-RU and using the presence of the augmented **ipv6** container in the o-ran-interfaces YANG module to indicate IPv6 is supported.

The IP interface(s) used to support UDP/IP based C/U plane transport may be different than the IP interface(s) used to support management plane connectivity. When different IP interface(s) is/are used, the C/U plane IP interfaces shall be configured in the O-RU by the NETCONF client by using the ietf-ip YANG model to configure the **IPv4** container and/or **IPv6** container. When defined by the NETCONF client, this interface shall be configured using either a named Ethernet interface (i.e., where the interface **type** is set to **ianaift:ethernetCsmacd)** and/or a named VLAN interface (i.e., where the interface **type** is set to **ianaift:l2vlan)**, depending upon whether VLANs are used to support IP based C/U plane traffic.

When a separate C/U plane IP interface is configured by the NETCONF client, additionally the NETCONF client may statically configure the IP address(es) on this/these interface(s). If the NETCONF client does not statically configure an IP address, the O-RU shall be responsible for performing IP address assignment procedures on the configured interfaces.

When an O-RU has not been configured with a static IP address, the O-RU shall support the IP address assignment using the following techniques:

1. IPv4 configuration using DHCPv4 [10].

And when the O-RU supports IPv6:

2. IPv6 Stateless Address Auto-Configuration (SLAAC) [11].

3. IPv6 State-full address configuration uses DHCPv6 [12].

4.5 Definition of processing elements

The CU-plane application needs to be uniquely associated with specific data flows. This association is achieved by defining an O-RU "processing element" which can then be associated with a particular C/U plane endpoint address [2] or delay measurement operation.

The O-RU management plane supports different options for defining the transport-based endpoint identifiers used by a particular processing element (used depending on transport environment), supporting the following 3 options:

- Processing element definition based on usage of different (alias) MAC addresses;
- Processing element definition based on a combination of VLAN identity and MAC address; and
- Processing element definition based on UDP-ports and IP addresses.

  Note: there is no well-defined source port currently allocated by IANA for the o-ran application and hence the NETCONF client is responsible for configuring this port number in the O-RU.

A processing element defines both the local and remote endpoints used with a specific data flow. The processing element definition includes its element **name** which is then used by other systems to refer to a particular processing element instance.

The o-ran-interfaces YANG model is used to define feature support for C/U plane transport based on alias MAC addresses and UDP/IP. The exchange of NETCONF capabilities is used to signal which optional capabilities are supported by the O-RU, as described in Annex C.

The o-ran-processing-elements YANG model uses a **processing-elements** container to define a list of processing elements. Each processing element is identified by a unique element **name**. Each processing element references a particular **interface-name** used to support the data flows associated with a particular processing element. Depending upon the type of C/U plane transport session, additionally leafs are configured that specify MAC addresses, and/or VLANs and/or IP addresses and/or UDP ports used to identify a particular processing element.

4.6 Verifying C/U Plane Transport Connectivity

As described above, there will likely be multiple C/U-plane data flows being exchanged between the O-DU and the O-RU. In order to enable checks verifying end-to-end transport connectivity between the O-DU and O-RU, the O-RU shall support transport connectivity check capabilities using a request/reply function.

Using that connectivity monitoring procedure, reachability/connectivity checks between user plane endpoints can be performed by the O-DU:

- During O-RU configuration, to validate the transport configuration
- At runtime to monitor network connectivity

In packet networks connectivity checking is usually done by exchanging probe-messages between the endpoints. The periodicity of this exchange depends on the use case. For availability measurement, the only use case relevant for this specification, the periodicity is usually between 1 and 60 seconds.

Two different network protocols are defined for performing the transport connectivity check procedure:

- For C/U sessions over Ethernet: Loop-back Protocol (LB/LBM) as defined by IEEE 802.1Q (amendment 802.1ag) [17].
- When the O-RU supports C/U sessions over IP: UDP echo, RFC 862 [18].

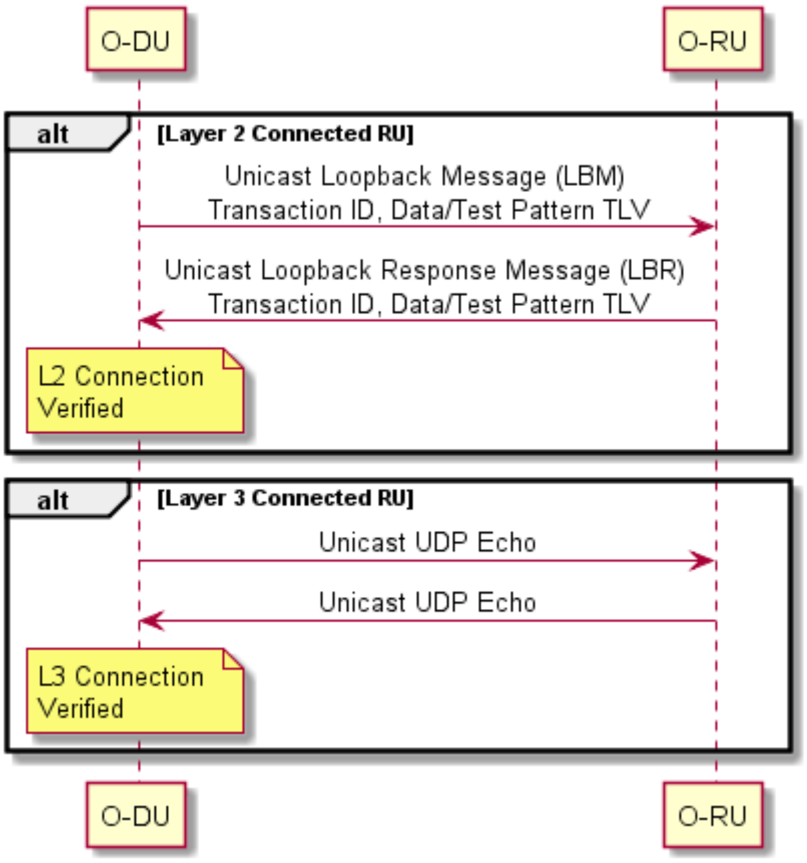

O-DU
O-RU
alt
[Layer 2 Connected RU]
Unicast Loopback Message (LBM)
Transaction ID, Data/Test Pattern TLV
Unicast Loopback Response Message (LBR)
Transaction ID, Data/Test Pattern TLV
L2 Connection Verified
alt
[Layer 3 Connected RU]
Unicast UDP Echo
Unicast UDP Echo
L3 Connection Verified
O-DU
O-RU

Figure 8: C/U Plane Transport Connectivity Verification

4.6.1 Ethernet connectivity monitoring procedure

If the O-RU and O-DU are operating their C/U sessions on Ethernet, the transport connectivity verification checks operate at the Ethernet layer. In this O-RU Management Plane Specification, the protocol for Ethernet connectivity monitoring is based on the Loop-back Protocol as defined by IEEE 802.1Q (amendment 802.1ag) [17].

For the purpose of connectivity monitoring all C/U -plane messaging endpoints in the fronthaul network are part of the same Maintenance Entity (ME). They each get the assigned the role of a Maintenance association End Point (MEP) for LBM.

The sending of Loop-back Messages (LBMs) is administratively initiated and stopped in the O-DU. Therefore, sending LBM requests needs to be requested by an administration entity, specifying an Ethernet MAC address assigned to the O-RU responder. In this O-RU Management Plane Specification requests are always sent from an O-DU to a unicast Ethernet MAC address of an O-RU.

4.6.1.1 Validating the transport configuration

After setting up a U/C-plane session between an O-DU and an O-RU, the O-DU can test whether connectivity exists as per the configuration. To achieve that, at the time a U/C-plane messaging endpoint becomes operational at an O-RU, it starts an LBM responder application which automatically responds to incoming LBM requests on that endpoint. Based on a configuration command the O-DU starts sending out a predefined number of LBM requests to its O-RU(s) at a predefined interval, storing the information received in LBM responses from the O-RU(s) in an internal database. O-RU(s) are identified by both Ethernet MAC address and the CU plane VLAN.

Note, the O-RU shall be able to respond to LoopBack Messages received from different remote Maintenance Association Endpoints

In case the configuration of the session is indeed correct, the O-DU should receive LBM responses from the O-RU(s) within a time frame dependent on the network latency and the O-RU's reaction time. If LBMs from the O-RU(s) are being received, the session is determined to be operational.

4.6.1.2 Monitor network connectivity

After the procedure described in section 4.6.1.1 has been executed successfully, a further procedure may be executed continuously to maintain the connectivity status. To achieve this the O-DU continues to send out LBM requests at the configured interval. It also keeps track of LBM responses received.

Based on the LBM responses received the O-DU shall decide on the connectivity status. Connectivity shall be assumed to be available as long as LBM responses from the O-RU(s) are being received at the configured interval. Connectivity shall be assumed not available if no LBM response from the particular O-RU has been received for an interval that is as long as 3 x the configured LBM request interval or longer.

4.6.1.3 Managing Ethernet connectivity monitoring procedure

An O-DU may have one or more Ethernet interfaces that have to support the Ethernet connectivity monitoring procedure. This section describes the management of this function. The module described here is based on (i.e. a subset of) the mef-cfm module defined by the Metro Ethernet Forum [19]. This is to allow for a later extension of the module to the full feature set of mef-cfm.

The YANG module provided below supports the configuration and fault management of the Loop-back Protocol as defined by IEEE 802.1Q (amendment 802.1ag).

Derived from MEF CFM YANG, the subset of type definitions are defined as part of the o-ran-lbm.yang.

4.6.2 IP connectivity monitoring procedure

If the O-RU and O-DU are connected using IP (and UDP/IP is being used to transport the C/U plane), these transport connectivity verification checks operate at layer 3. Layer 3 connection verification is based on the O-RU supporting the UDP echo server functionality, RFC 862 [18]. The NETCONF client is responsible for enabling the UDP echo server in the O-RU, triggering the O-RU to listen for UDP datagrams on the well-known port 7. When a datagram is received by the O-RU, the data from it is sent back towards the sender, where its receipt can be used to confirm UDP/IP connectivity between the endpoints.

4.6.2.1 Managing IP Connectivity Monitoring Procedure

This section describes the management of the UDP echo functionality. The NETCONF client uses the **enable-udp-echo** leaf in the udp-echo YANG model to control operation of the UDP echo server in the O-RU. The NETCONF client is able to control the DSCP marking used by the O-RU when it echoes back datagrams using the **dscp-config** leaf. Additionally, the NETCONF client can recover the number of UDP Echo messages sent by the O-RU by using **echo-replies-transmitted** operational state.

An O-DU may have one or more IP interfaces that have to support the UDP/IP connectivity monitoring procedure. An O-RU with its UDP echo server enabled shall be able to respond to UDP datagrams originated from any valid source IP address.

4.7 C/U-Plane Delay Management

The Intra-PHY lower layer fronthaul split has the characteristic of a stringent bandwidth and tight latency requirement. The CUS-Plane specification [2] describes how the propagation delay incurred due to distance between the O-DU and O-RU is an important parameter in defining the optimization of windowing and receive-side buffering operations. This section describes the procedures that are used to manage the delay parameters for the fronthaul split.

4.7.1 Delay Parameters

The reference points for delay management are introduced in [2] and included for reference in Figure 9.

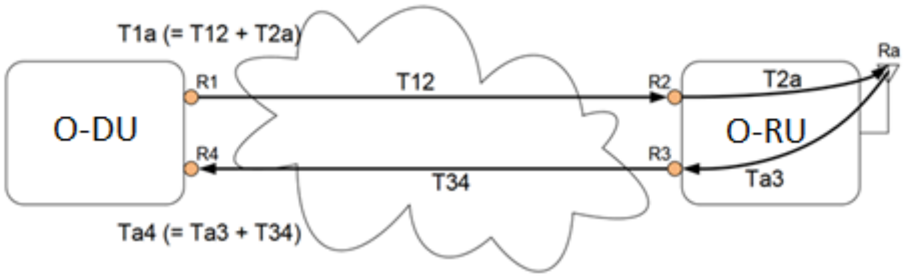


Figure 9: Definition of reference points for delay management

Important delay parameters related to the operation of the O-RU are referred to as the O-RU delay profile. As the delay characteristics for an O-RU may vary based on air interface properties, a table of the parameters is provided based on a combination of sub-carrier spacing (SCS) and channel bandwidth. When considering the downlink data direction, these parameters include:

T2a_min : Corresponding to the minimum O-RU data processing delay between receiving the last data sample over the fronthaul interface and transmitting the first IQ sample at the antenna.

T2a_max: Corresponding to the earliest allowable time when a data packet is received before the corresponding firs IQ sample is transmitted at the antenna.

Using the above parameters, (T2a_max – T2a_min): The difference between these two parameters corresponds to the O-RU reception window range.

T2a_min_cp_dl: Corresponding to the minimum O-RU data processing delay between receiving downlink real time control plane message over the fronthaul interface and transmitting the corresponding first IQ sample at the antenna.

T2a_max_cp_dl: Corresponding to the earliest allowable time when a downlink real time control message is received before the corresponding first IQ sample is transmitted at the antenna.

Tcp_adv_dl: Corresponding to the time difference (advance) between the reception window for downlink real time Control messages and reception window for the corresponding IQ data messages.

The delay parameters related to the operation of the O-RU for the uplink data direction include:

Ta3_min: Corresponding to the minimum O-RU data processing delay between receiving an IQ sample at the antenna and transmitting the first data sample over the fronthaul interface.

Ta3_max: Corresponding to the maximum O-RU data processing delay between receiving an IQ sample at the antenna and transmitting the last data sample over the fronthaul interface.

Using the above parameters, (Ta3_max – Ta3_min): The difference between these two parameters corresponds to the O-RU transmission window range.

T2a_min_cp_ul: The minimum O-RU data processing delay between receiving real time up-link control plane message over the fronthaul interface and receiving the first IQ sample at the antenna.

T2a_max_cp_ul: The earliest allowable time when a real time up-link control message is received before the corresponding first IQ sample is received at the antenna.

When requested, all O-RUs shall signal the table of statically “pre-defined” values of the above parameters, for different supported combinations of SCS and channel bandwidth over the management plane interface. This will typically occur during the initial startup phase.

4.7.2 Reception Window Monitoring

The O-RU shall monitor operation of its reception window, monitoring the arrival of packets received over the fronthaul interface relative to the earliest and latest allowable times as defined by the values of T2a_max and T2a_min respectively.

See Annex B.2 for information on reception window counters.

4.8 O-RU Adaptive Delay Capability

O-RUs may optionally support the ability to optimize their buffers based on information signalled concerning the configuration of the O-DU, e.g., including the O-DU delay profile, together with transport delay information, which may have been derived by the O-DU by using a delay measurement procedures operated by the O-DU or by other techniques. This section describes such optional O-RU buffer optimization functionality.

An O-RU that supports the optional adaptive timing capability shall indicate such to the O-RU Controller client by exchanging NETCONF capabilities, as described in Section 6.2 and Annex C indicating that it supports the ADAPTIVE-O-RU-PROFILE feature. An O-RU Controller may then provide the O-RU with the O-DU delay profile based on a combination of sub-carrier spacing (SCS) and channel bandwidth, comprising the following parameters:

T1a_max_up: Corresponding to the earliest possible time which the O-DU can support transmitting an IQ data message prior to transmission of the corresponding IQ samples at the antenna

TXmax: Corresponding to the maximum amount of time which the O-DU requires to transmit all downlink user plane IQ data message for a symbol.

Ta4_max: Corresponding to the latest possible time which the O-DU can support receiving the last uplink user plane IQ data message for a symbol

RXmax: Corresponding to the maximum time difference the O-DU can support between receiving the first user plane IQ data message for a symbol and receiving the last user plane IQ data message for the same symbol.

In addition to the O-DU delay profile, the O-RU-Controller provides the O-RU with the transport network timing parameters:

T12_min: Corresponding to the minimum delay between any O-DU and O-RU processing elements

T12_max: Corresponding to the maximum delay between O-DU and O-RU processing elements

T34_min: Corresponding to the minimum delay between any O-RU and O-DU processing elements

T34_max: Corresponding to the maximum delay between O-RU and O-DU processing elements

The O-RU may use this information to adapt its delay profile, ensuring that the inequalities defined in Annex B of [2] are still valid

The O-RU controller will typically provide this information during the O-RU's start-up procedure. If an O-RU receives the adaptive delay configuration information when operating a carrier, the O-RU shall not adapt its O-RU delay profile until all carriers operating using the O-RU buffers have been disabled. Once an O-RU has adapted its O-RU profile, it will include the newly adapted timing values when signaling its delay parameters to a NETCONF client.

4.9 Measuring transport delay parameters

An O-RU may optionally indicate that it supports the eCPRI based measurement of transport delays between O-DU and O-RU, as described in [2].

When indicated that it supports the optional eCPRI based delay measurement, an O-RU shall support the processing of delay measurement messages at any time, including whenever the C/U plane is configured. As well as the O-RU including its timing compensation values in the eCPRI delay measurement messages (tcv1 and tcv2 shown in the figure below), the YANG model additionally enables a NETCONF client to recover these parameters from the O-RU over the M-Plane interface.

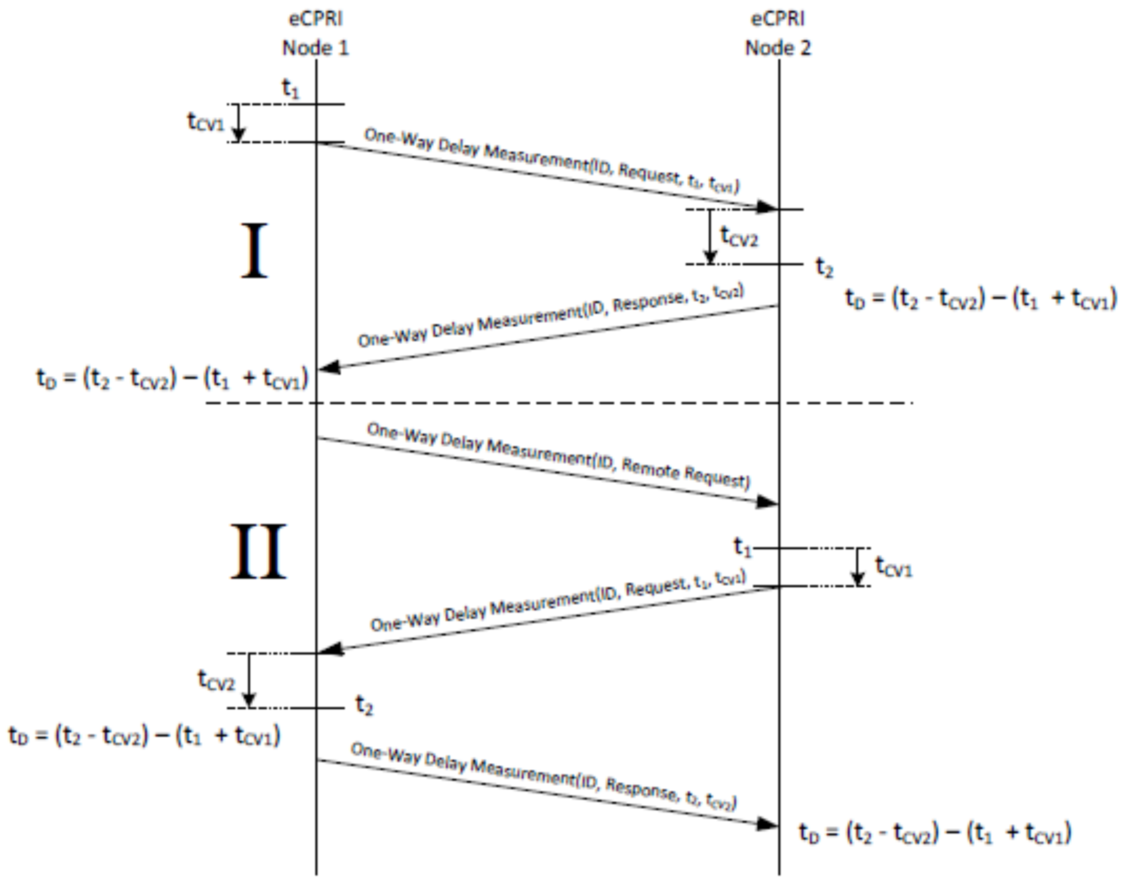

eCPRI Node 1
eCPRI Node 2
t1
tCV1
One-Way Delay Measurement(ID, Request, t1, tCV1)
tCV2
t2
tD = (t2 - tCV2) – (t1 + tCV1)
I
One-Way Delay Measurement(ID, Response, t2, tCV2)
tD = (t2 - tCV2) – (t1 + tCV1)
One-Way Delay Measurement(ID, Remote Request)
II
t1
tCV1
One-Way Delay Measurement(ID, Request, t1, tCV1)
tCV2
t2
tD = (t2 - tCV2) – (t1 + tCV1)
One-Way Delay Measurement(ID, Response, t2, tCV2)
tD = (t2 - tCV2) – (t1 + tCV1)

Figure 10: eCPRI One-Way Delay Measurement procedure [2]

An O-RU that supports the eCPRI based delay measurement capability, shall be able to support simultaneous operation of delay measurements over any processing element configured as described in section 4.5. For each processing element configured for use by the delay measurement procedure, the O-RU shall keep a record of the number of responses, requests and follow-up messages transmitted by the O-RU.

Chapter 5 Software Management

The Software Management function provides a set of operations allowing the desired software build to be downloaded, installed and activated at O-RU. Successful software activation operation does not mean an O-RU is running the just activated software build. An O-RU **reset** RPC is required to trigger the O-RU to take the activated software build into operational use.

A single software build is considered as set of internally consistent files compliant within such a build. Replacement of files within a build is prohibited, as this will cause software version incompatibility. Software build is a subject of versioning and maintenance and as such cannot be broken.

The use of compression and ciphering for the content of the software build is left to vendor implementation. The only file which shall never be ciphered is the manifest.xml file.

It is also Vendor's responsibility to handle SW Build / package / file integrity check.

The O-RU provides a set of so called "software slots" or "slots". Each slot provides an independent storage location for a single software build. The number of slots offered by O-RU depends on the device's capabilities. At least two writable slots shall be available at the O-RU for failsafe update operation. Presence of read only slot is optional. The software slots are resources provided by the O-RU and as such are not the subject of creation and deletion. The size of individual software slots is fixed and determined by the O-RU's vendor and sufficient to accommodate the full software build.

Note: procedures used in Software Management are covered by o-ran.software-management.yang module.

5.1 Software Package

The software package is delivered by the O-RU vendor.

Each software package includes:

- manifest.xml
- software files to be installed on O-RU

The name of package should follow the following format:

"<Vendor Code><Vendor Specific Field>[#NUMBER].EXT"

Where:

- *Vendor Code* is a mandatory part which has two capital characters,
- *Vendor Specific Field* is any set of characters allowed in filename. The value must not include character "_" (underscore) or "#" (hash). The value can be defined per vendor for the human readable information. Version information is necessary in the *Vendor Specific Field* which defines load version,
- *NUMBER* is optional and used when the original file is split into smaller pieces – number after "#" indicates the number of a piece. Numbering starts from 1 and must be continuous,
- *EXT* is a mandatory part which defines the extension of filename. A vendor provides one or more files which may be compressed (.ZIP) or uncompressed.

Note: the name of package needs a <Vendor Code> prefix to avoid issue if two vendors provide files with same name.

Note: the NETCONF client shall support ZIP functionality to enable support of compressed files types.

Note: the operator needs to manage and control which O-RU files will be stored and used in the file server, e.g., based on O-RU files provided by O-RU vendors and network configurations. The operator needs to make sure that only the expected version of O-RU files will be transferred from the file server to the O-RU. Different versions of files for O-RU with the same vendor and same product code should be avoided.

The content of the manifest.xml file allows to maintain software update process correctly in terms of compatibility between O-RU hardware and software build to be downloaded. The content of the Manifest file prohibits the O-RU from installing software builds designed for device based on different hardware. The format of the manifest.xml file is:

```
<xml>
<manifest version="1.0"> /// @version describes version of file format (not the content)
   <products>
      <product vendor="XX" code="0818820\.x11" name="RUXX.x11" build-Id="1"/>
      <product vendor="XX" code="0818820\.x12" name="RUXX.x12" build-Id="1"/>
      <product vendor="XX" code="0818818\…" name="RUYY" build-Id="2"/>
      /// @vendor is as reported by O-RU
      /// @code is a regular expression that is checked against productCode reported by O-RU
      /// @name is optional and used for human reading - MUST NOT be used for other purposes!
      /// @buildId is value of build@id (see below)
   </products>
   <builds>
      <build id="1" bldName="xyz" bldVersion="1.0">
      /// @id is index of available builds.
      /// @bldName and @bldVersion are used in YANG (build-name, build-version)
         <file fileName="xxxx" fileVersion="1.0" path="full-file_name-with-path-relative-to-
package -root-folder" checksum="FAA898"/>
         <file fileName="yyyy" fileVersion="2.0" path="full-file_name-with-path-relative-to-
package -root-folder" checksum="AEE00C"/ >
         /// @fileName and @fileVersion are used in YANG (name, version)
         /// @path is full path (with name and extension) of a physical file, relative to
package root folder,used in YANG (local-path)
         /// @checksum is used to chech file integrity on O-RU side
      </build>
      <build id="2" bldName="xyz" bldVersion="1.0">
         <file fileName="xxxx" fileVersion="1.0" path="full-file_name-with-path-relative-to-
package -root-folder" checksum="FAA898"/>
```

```
            <file fileName="yyyy" fileVersion="2.0" path="full-file_name-with-path-relative-to-
package -root-folder" checksum="AEE00C"/ >
            <file fileName="zzzz" fileVersion="1.5" path="full-file_name-with-path-relative-to-
package -root-folder" checksum="ABCDEF"/ >
        </build>
    </builds>
</manifest>
</xml>
```

Note: Keywords in manifest.xml example are in **bold**, the keywords must be strictly followed considering of cross-vendor cases.

Note 2: Correspondency between content of manifest.xml tags and content of o-ran-software-management.yang is:

- XML tag "product vendor" corresponds to content leaf "vendor-code",

- XML tag "code" corresponds to content of leaf "product-code",

- XML tag "build-Id" corresponds to content of leaf "build-id",

- XML tag "bldName" corresponds to content of leaf "build-name",

- XML tag "bldVersion" corresponds to content of leaf "build-version",

- XML tag "fileName" corresponds to content of leaf "name" in list "files",

-XML tag "fileVersion" corresponds to content of leaf "version" in list "files"

5.2 Software Inventory

Pre-condition:

- M-Plane NETCONF session established.

Post-condition:

- NETCONF client successfully collected the software inventory information from NETCONF server.

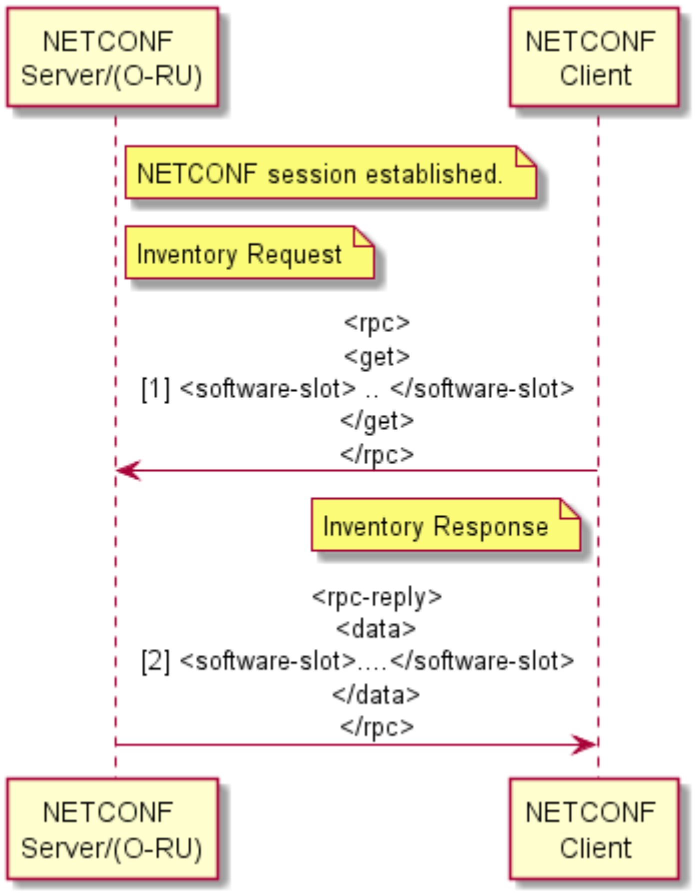


Figure 11: Inventory fetch call flow

Software Inventory is fetched by a NETCONF Client using the NETCONF **get** rpc filtered over the **software-slot** container. The response contains information about each software slot and its contents.

The following information is provided by software-inventory reply message:

a) **name** - name of the software slot (the name is defined by the O-RU vendor)

b) **status** - status of the software package. Status of the package can be

- **VALID** - Slot contains a software build considered as proven valid.

- **INVALID** - software build is not currently used by O-RU. The software is considered by the O-RU as damaged (e.g. wrong CRC).

Activation of a software slot containing an invalid software build shall be prohibited. Note: failed software install operation can cause a slot status to change to "Invalid".

- **EMPTY** - software slot does not contain a software package. Activation of an empty software slot shall be prohibited.

c) **active** - indicates if the software stored in particular slot is activated at the moment.

- True - software slot contains an activated software build. Active::True can be assigned only for slots with status "Valid". At any time, only one slot in the O-RU MUST be marked as Active::True. The O-RU shall reject activation for software slots with status "Empty" and "Invalid".

- False - software slot contains passive software build or is empty

d) **running** - informs if software stored in particular slot is used at the moment.

- **True** - software slot contains the software build used by the O-RU in its current run.

- **False** - software slot contains a software build not used by O-RU at the moment.

e) **access** – informs about access rights for the current slot

- **READ_ONLY** – The slot is intended only for factory software, activation of such software slot means performing a factory reset operation and a return to factory defaults settings.
- **READ_WRITE** – slot used for updating software

f) **product-code** - product code provided by the vendor, specific to the product.

g) **vendor-code** - unique code of the vendor.

h) **build-id** - Identity associated with the software build. This id is used to find the appropriate build-version for the product consist of the vendor-code and the product-code.

i) **build-name** - Name of the software build.

j) **build-version** - Version of the software build for the product consist of the vendor-code and the product-code.

k) **files** – list of files in build

l) **name** – name of one particular file

m) **version** – version of the file

n) **local-path** - complete path of the file on local file system

o) **integrity** - result of the file integrity check

- **OK** – file integrity is correct
- **NOK** – file is corrupted

If a slot contains a file with integrity::NOK, the O-RU shall mark the whole slot with status::INVALID. The content of a software-slot is fully under O-RU's management - including removal of the content occupying the slot (in case the slot is subject of software update procedure), control of file system consistency and so on. The slot content shall not be removed until there is a need for new software to be installed.

Note: The empty slot parameters shall be as follows

name: up to vendor, not empty

status: "INVALID"

active: False

running: False

access: READ_WRITE

product-code: up to vendor

vendor code: up to vendor

build-name: null (empty string)

build-version: null (empty string)

files: empty

5.3 Download

Pre-condition:

- M-Plane NETCONF session established.

Post-condition:

- O-RU downloads all files specified and successfully stores the downloaded files in the O-RU's file system.

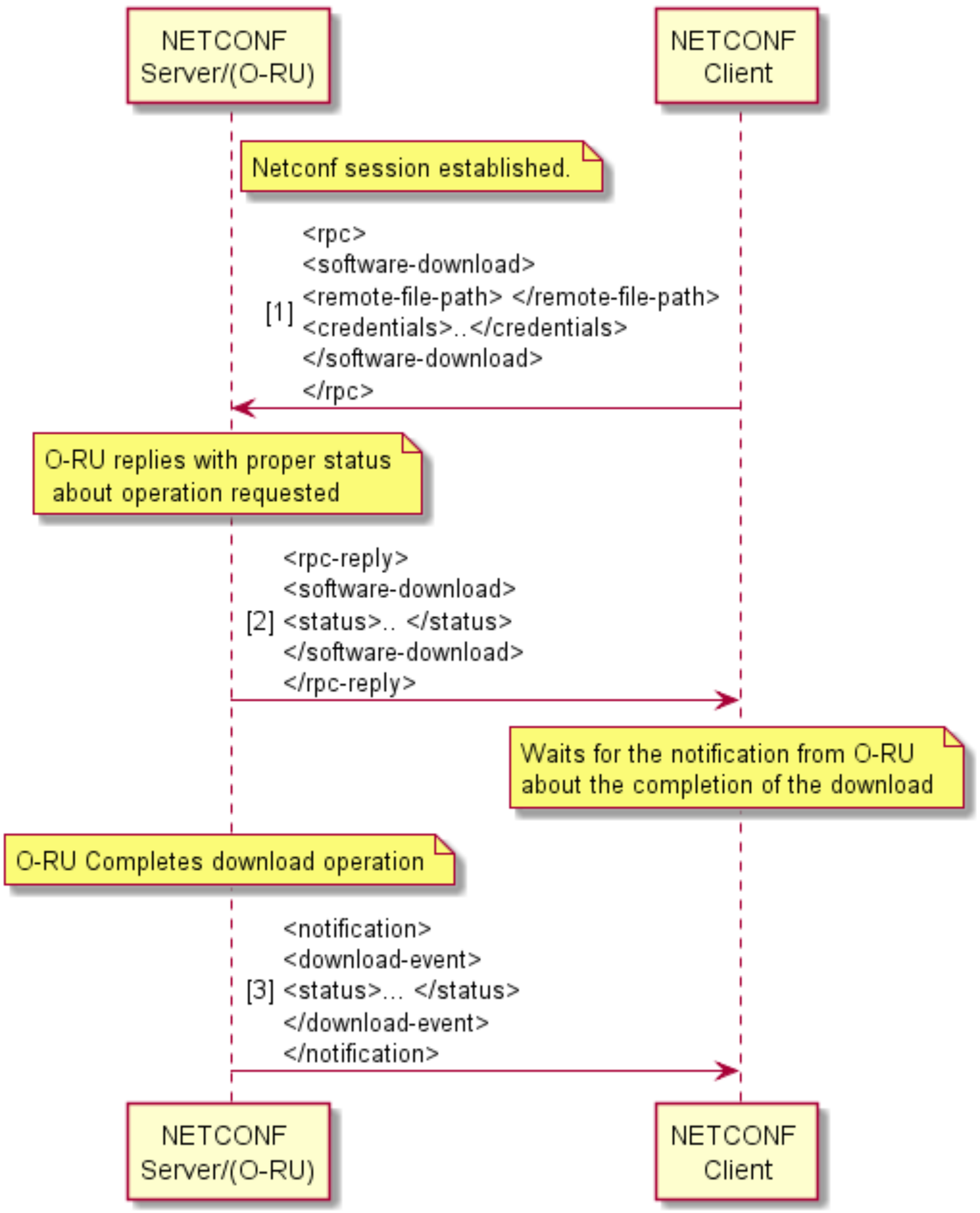


Figure 12: Software download call flow

Following types of authentications are supported for **software-download**:

a) password

b) certificate

The **software-download** rpc is used to trigger the downloading of software to the O-RU. The download shall be performed using sFTP. The rpc specifies the URI of the remote location of the software files.

The O-RU shall send an immediate rpc-reply message with one of following statuses:

a) STARTED – software download operation has been started

b) FAILED – software download operation could not be proceeded, reason for failure in error-message

When the O-RU completes the software download or software download fails, the O-RU shall send NETCONF **download-event** notification with one of the following statuses:

a) COMPLETED

b) AUTHENTICATION_ERROR - source available, wrong credentials

c) PROTOCOL_ERROR – sFTP protocol error

d) FILE_NOT_FOUND - source not available

e) APPLICATION_ERROR - operation failed due to internal reason

f) TIMEOUT - source available, credentials OK, Operation timed out (e.g. source becomes unavailable during ongoing operation).

The above will be repeated until all files which are required for the O-RU and which belong to the software package have been downloaded to the O-RU.

5.4 Install

Pre-condition:

- M-Plane NETCONF session established.
- At least one software slot with status active::False and running::False exists in O-RU.
- Software Download has been completed successfully and files are available in O-RU

Post-condition:

- O-RU software is installed in the specified target **software-slot**.

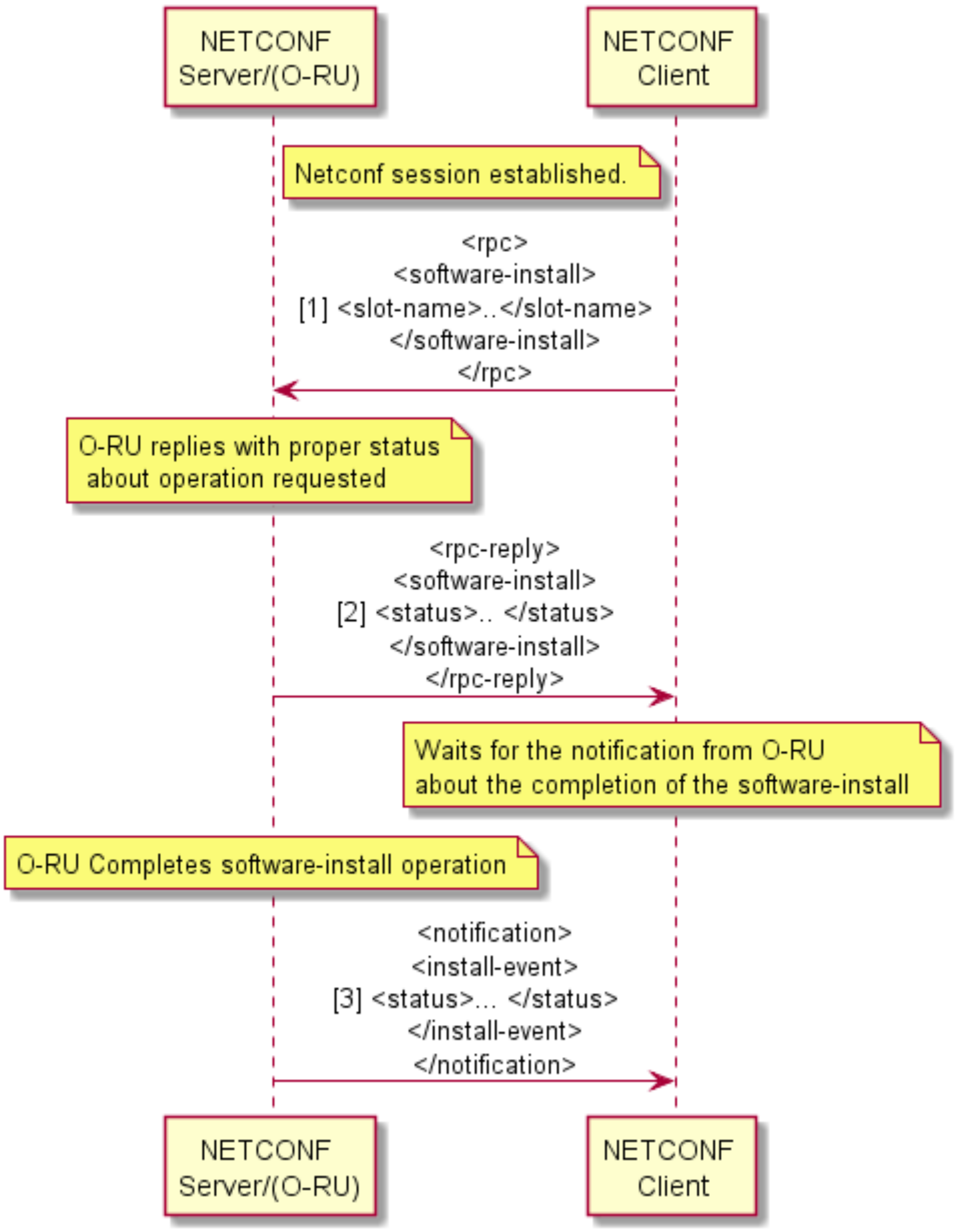


Figure 13 Software install call flow

NETCONF **software-install** rpc is used to install the previously downloaded software (all files provided in the package) to the specified target **software-slot** on O-RU**.** This slot must have status active::False and running::False.

The O-RU shall send an immediate rpc-reply message with one of following statuses:

a) STARTED – software install operation has been started.

b) FAILED – software install operation could not be proceeded, reason for failure in error-message.

When O-RU completes the software install or software install procedure fails, the O-RU will send NETCONF **install-event** notification with one of the following statuses:

a) COMPLETED - Install procedure is successfully completed.

b) FILE_ERROR – operation on the file resulted in in error, disk failure, not enough disk space, incompatible file format

c) INTEGRITY_ERROR – file is corrupted

d) APPLICATION_ERROR – operation failed due to internal reason

When the software install commences, the O-RU shall set the slot status to INVALID. After the install procedure finishes, the O-RU shall change the slot status to its appropriate status. This operation avoids reporting of inaccurate status when the install procedure is in operation or when it is interrupted (e.g., by spurious reset operation).

5.5 Activation

Pre-condition:

- M-Plane NETCONF session established.
- Software slot to be activated has status VALID.

Post-condition:

- O-RU software activates to the version of software-slot.

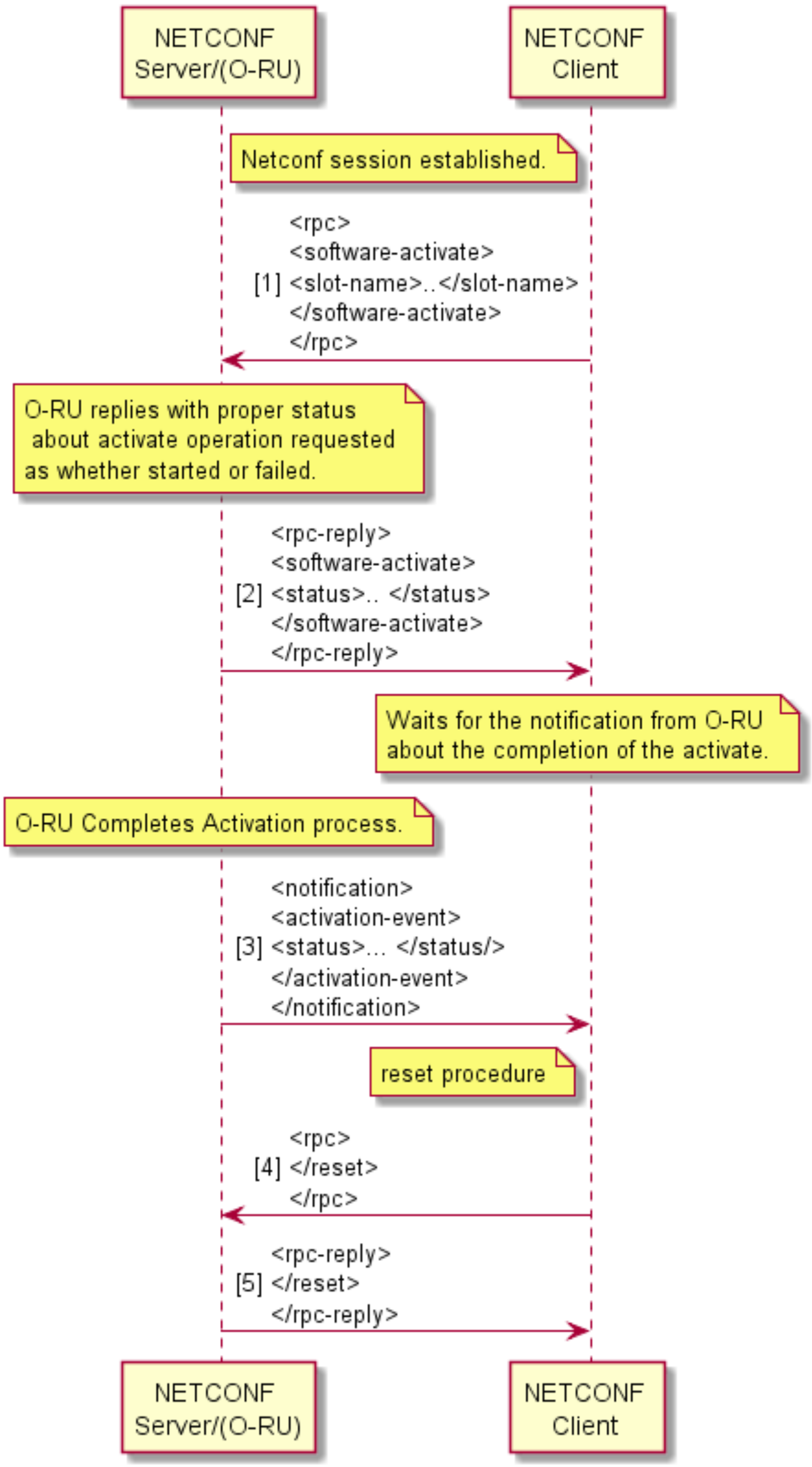


Figure 14 Software activation call flow.

NETCONF **software-activate** rpc is used to activate the software. The name of the software-slot is specified in the activate request.

The O-RU shall send an immediate rpc-reply message with one of following statuses:

a) STARTED – software activation operation has been started

b) FAILED – software activation operation could not be proceeded, reason for failure in error-message

When the activation is completed, the O-RU shall send the NETCONF **activation-event** notification with the status of activation. The following status is returned in the NETCONF **activation-event** notifications.

a) COMPLETED - Activation procedure is successfully completed. O-RU must be restarted via NETCONF **reset** rpc for the new software to be activated.

b) APPLICATION_ERROR - operation failed due to internal reason

Only one software slot can be active at any time. Thus, successful software-activate command will set active::True to the slot that was provided in the rpc and automatically will set active::False to the previously active slot.

NETCONF **reset** rpc shall be sent to O-RU to activate to the version of the software-slot. O-RU restarts and performs startup procedure as described in Chapter 3 as regular startup with new software version running.

5.6 Software update scenario

An example scenario of a successful software update procedure can be as follows:

1. NETCONF client performs a software inventory operation and determines that a new software package is available and can be installed on the O-RU
2. NETCONF client using the **software-download** rpc requests the O-RU to download a software package (if the software package contains several files, steps 2-4 need to be performed repeatedly until all files have been downloaded)
3. O-RU sends rpc response that download was started
4. O-RU finishes downloading the file(s) and reports this by sending the **download-event** notification
5. NETCONF client requests installation of the software using **software-install** rpc, and provides the slot name where the software needs to be installed along with a list of filenames to be installed (if the software package contains only one file, the list will contain only one entry)
6. O-RU sends rpc response that installation was started
7. O-RU sets installation slot status to INVALID
8. O-RU installs the software and after successful installation (with checksum control) changes status of the slot to VALID
9. O-RU notifies the NETCONF client that the installation is finished using **install-event** notification
10. NETCONF client requests the O-RU to activate the newly installed software using the **software-activate** rpc
11. O-RU sends rpc response that activation was started
12. For requested slot, O-RU changes active to True and at the same time sets activate to False on previously active slot
13. O-RU notifies NETCONF client about activation finished using the **activation-event** notification
14. NETCONF client restarts the O-RU forcing it to use the newly installed and activated software into use. O-RU restarts as regular startup with new software version running.

5.7 Factory Reset

O-RU can be reset to the factory default software by activating the software-slot containing the factory default software and initiating NETCONF **reset** rpc. O-RU may clear persistent memory data during factory reset as vendor implementation option.

Chapter 6 Configuration Management

6.1 Baseline configuration

This chapter describes NETCONF standard operation (edit-config/get-config/get) [3] which belongs to the CM in Module to modify/retrieve any parameters in YANG modules. Examples below use o-ran-hardware as an example YANG module.

Two following scenarios are feasible for Configuration Management purposes.

- 2 phase (modify/commit) operation using writable running datastore
- 3 phase (modify/commit/confirm) operation using candidate datastore.

The 2-phase operation performs edit config on running datastore directly and confirm operation is not used.

All O-RUs shall support 2 phase operation, with 3 phase operations being optional.

6.1.1 Retrieve State

O-RU Controller is able to retrieve state which is defined in o-ran-hardware by using NETCONF <get-config> or <get> procedure:

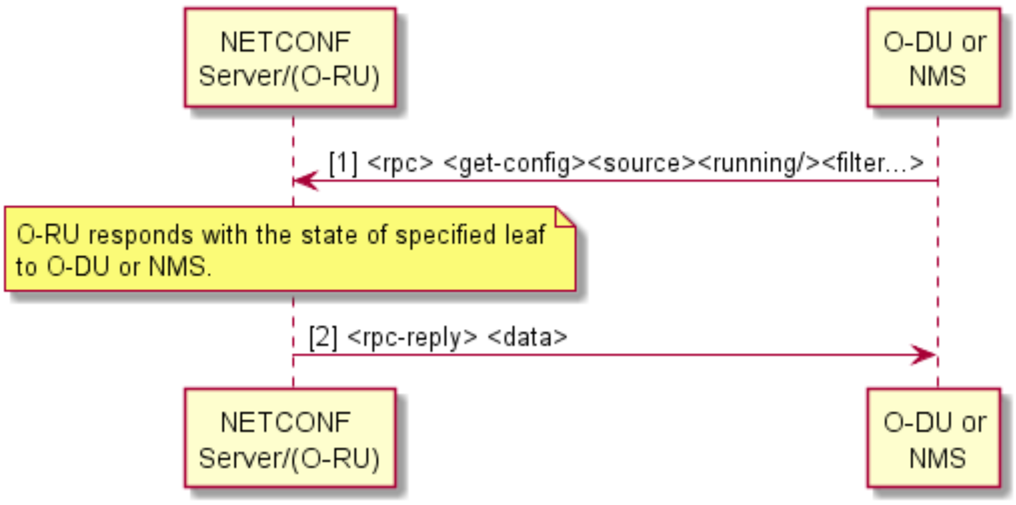

NETCONF Server/(O-RU)
O-DU or NMS
[1] <rpc> <get-config><source><running/><filter...>
O-RU responds with the state of specified leaf to O-DU or NMS.
[2] <rpc-reply> <data>
NETCONF Server/(O-RU)
O-DU or NMS

Figure 15 – Retrieve Resource State

Preconditions:

- O-RU Controller has completed exchange of NETCONF capabilities as part of connection establishment between O-RU and O-RU Controllers.

Post conditions:

- O-RU controller has retrieved O-RU state as per <get-config> or <get> request.

6.1.2 Modify State

O-RU Controller is able to change state which can be configurable by using NETCONF <edit-config> procedure without reset.

The configurable state are admin-state and power-state defined in o-ran-hardware.

In case of a failure, an error will be returned. Please refer to RFC6241 Appendix A for error codes.

The vendor can define the behavior after the error occurred.

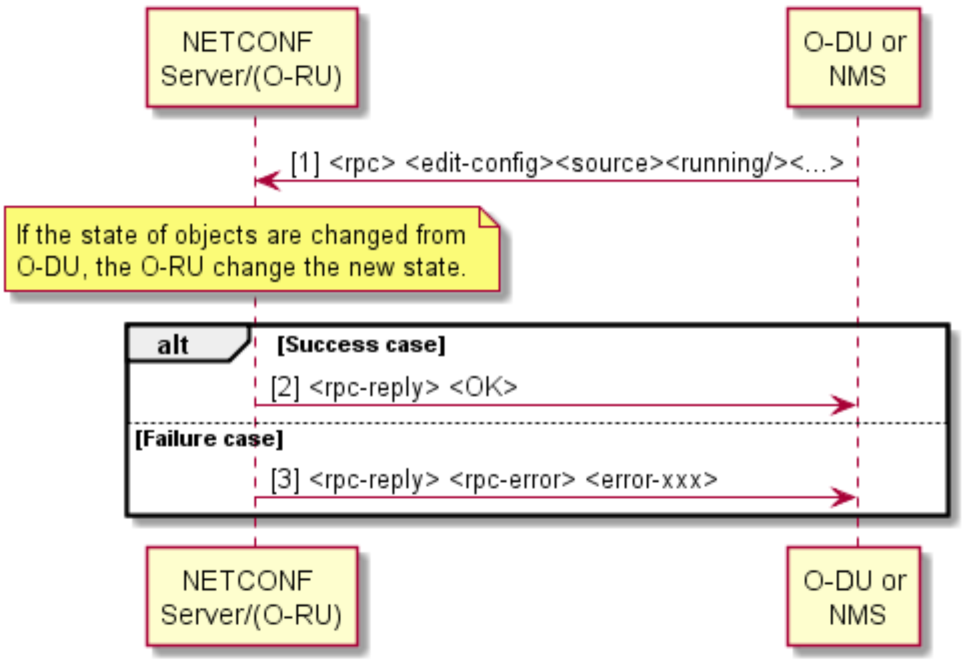

NETCONF Server/(O-RU)
O-DU or NMS
[1] <rpc> <edit-config><source><running/><...>
If the state of objects are changed from O-DU, the O-RU change the new state.
alt
[Success case]
[2] <rpc-reply> <OK>
[Failure case]
[3] <rpc-reply> <rpc-error> <error-xxx>
NETCONF Server/(O-RU)
O-DU or NMS

Figure 16 – Modify Resource State without reset

The followings are the information of state transition for each state.

[admin-state]

The admin-state transition diagram for the O-RU is illustrated in Figure 17 below.

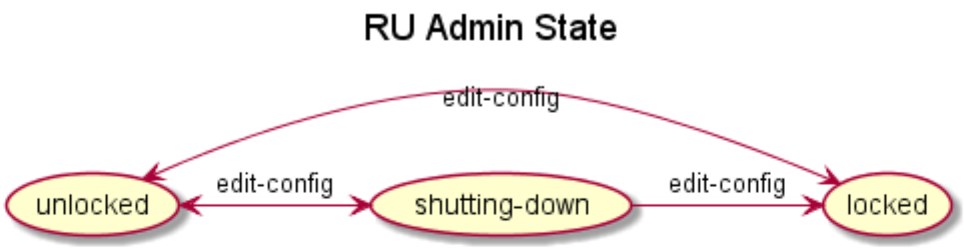


Figure 17 – Admin State

- locked: This state indicates that any resource activation is prohibited for the O-RU and all resources have been deactivated administratively.
- shutting-down: That usage is administratively limited to current instances of use. It's not mandatory (will be optional).
- unlocked: This state indicates that any resource activation is allowed and any resources can be active. The state "unlocked" is the initial state after the reset of the O-RU.

[power-state]

The power-state transition diagram for O-RU is presented in Figure 18 below. This state can be controlled by editing the parameter energy-saving-enabled.

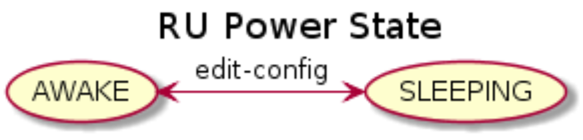


Figure 18 – Power State

- awake: This state indicates that the O-RU is operating normally, i.e. not in energy saving mode. The state "awake" is the initial state after the reset of the O-RU.
- sleeping: This state indicates that the O-RU is in energy saving mode. M-plane connection and functions are alive whereas other C/U/S functions may be stopped to reduce energy consumption. This state is optional.

[oper-state]

O-RU Controller is able to change oper-state defined in o-ran-hardware of the O-RU by using remote procedure call **reset**. In this case, the O-RU responds <rpc-reply><ok> prior to reset operation. Whatever the previous state is, the O-RU oper-state starts from disabled when O-RU receives **reset**.

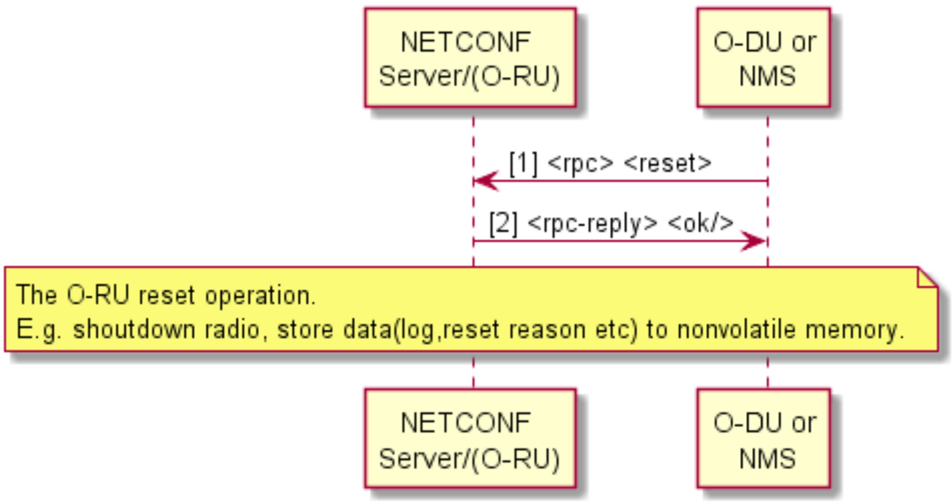

NETCONF Server/(O-RU)
O-DU or NMS
[1] <rpc> <reset>
[2] <rpc-reply> <ok/>
The O-RU reset operation.
E.g. shoutdown radio, store data(log,reset reason etc) to nonvolatile memory.
NETCONF Server/(O-RU)
O-DU or NMS

Figure 19 – Modify Oper State (reset)

The oper-state transition diagram for O-RU is presented in Figure 20 below.

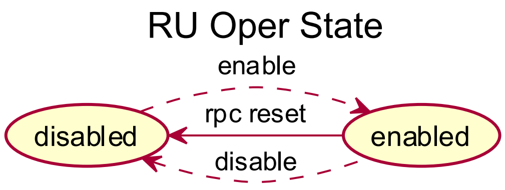


Figure 20 – Oper State

enabled: O-RU is partially or fully operational.

disabled: O-RU is not operational. This is the initial state of oper-sate after the reset of the O-RU

- O-RU Controller is able to reset the O-RU, even if the O-RU state is "disabled" or "enabled".

[availability-state]

The availability-state transition diagram for the O-RU is presented in Figure 21 below.

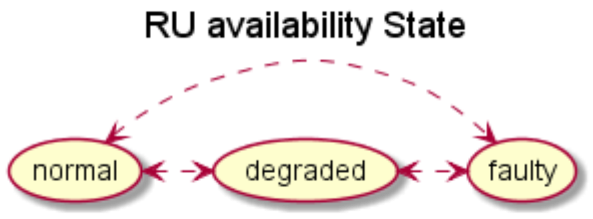


Figure 21: Availability State

The availability state is derived from detected and active faults and their impact to O-RU's operation. The availability state is not affected by faults caused by external reasons.

- normal: There is no fault.
- degraded: When major or critical fault affecting module or any of O-RU's subcomponents (e.g. transmitter) is active.
- faulty: The critical fault affecting whole O-RU is active and O-RU can't continue any services.

[usage-state]

The usage-state transition diagram for the O-RU is presented in Figure 22 below.

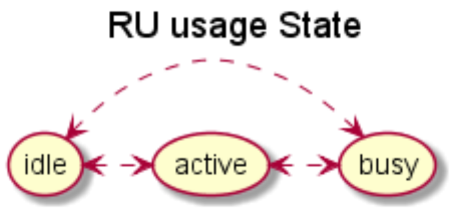


Figure 22 – usage State

idle: No carrier is configured in O-RU.

active: The carrier(s) is(are) configured in O-RU.

busy: No more carrier can be configured in O-RU.

6.1.3 Retrieve Parameters

O-RU Controller is able to retrieve parameters of the YANG module by using NETCONF <get-config> procedure

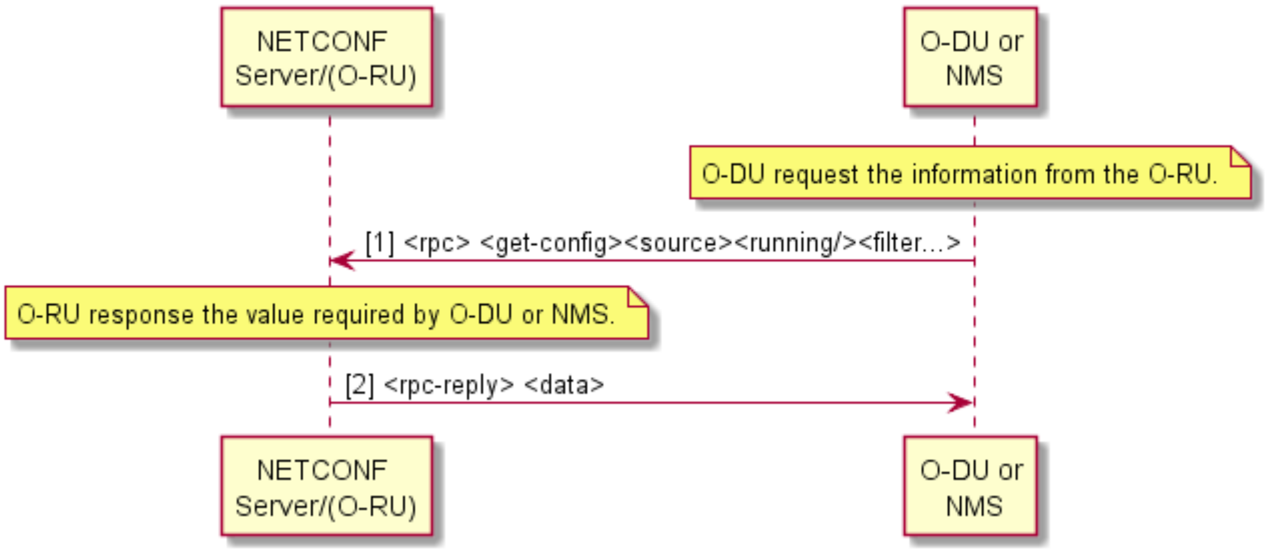

NETCONF Server/(O-RU)
O-DU or NMS
O-DU request the information from the O-RU.
[1] <rpc> <get-config><source><running/><filter...>
O-RU response the value required by O-DU or NMS.
[2] <rpc-reply> <data>
NETCONF Server/(O-RU)
O-DU or NMS

Figure 23 – Retrieve Parameters

Preconditions:

- O-RU Controller has completed exchange of NETCONF capabilities as part of connection establishment between O-RU and O-RU Controller(s).

Post conditions:

- O-RU controller has retrieved O-RU parameters as per <get-config><source><running/><filter> request.

6.1.4 Modify Parameters

O-RU Controller is able to modify parameters of the YANG module by using the NETCONF <edit-config> procedure.

In case of failures, an error will be returned. Please refer to RFC6241 Appendix A for error codes.

The vendor can define the behavior after error occurred.

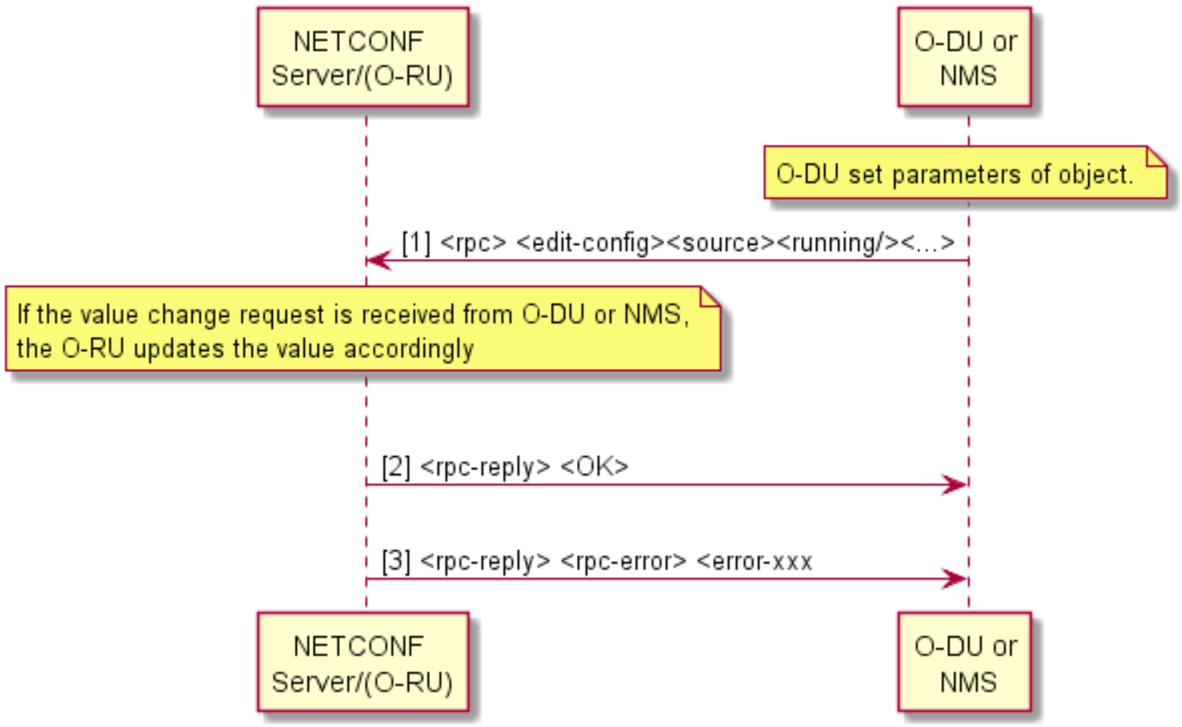

NETCONF Server/(O-RU)
O-DU or NMS
O-DU set parameters of object.
[1] <rpc> <edit-config><source><running/><...>
If the value change request is received from O-DU or NMS, the O-RU updates the value accordingly
[2] <rpc-reply> <OK>
[3] <rpc-reply> <rpc-error> <error-xxx
NETCONF Server/(O-RU)
O-DU or NMS

Figure 24 – Modify Parameters

Preconditions:

- O-RU Controller has completed exchange of NETCONF capabilities as part of connection establishment between the O-RU and O-RU Controller(s).

Post conditions:

- O-RU controller has retrieved O-RU resource state as per <edit-config> request

- Success case: The update is confirmed to O-RU Controller.
- Failure case: Failure reason is provided to O-RU Controller

Sequential processing is assumed. Only a single <edit-config> rpc is allowed at a time. Next <edit-config> rpc shall be performed after previous <edit-config> rpc reply.

Modify Parameters is used to:

- update parameters of existing leafs,

O-RU shall be allowed to reject <edit-config> in case the content is found to be against e.g., functions supported by O-RU - like carrier configured out of band.

6.2 Framework for optional feature handling

This section describes the common and optional features about Configuration Management.

An O-RU may have some features which are not supported by other O-RUs, i.e. optional feature(s). In this case, the O-RU needs to inform the O-RU controller which features the O-RU can provide, and this can be achieved by exchanging NETCONF capabilities.

Some of the YANG models are optional for the O-RU to support. For example, in this version of the management plane specification, those models associated with External IO and Antenna Line Devices are not essential for the operation of the O-RAN fronthaul interface. Other mandatory models define optional feature capabilities.

The NETCONF Server shall use the ietf-yang-library model (RFC 7895) [20] to list the namespace of the models supported by the Server. If an O-RU/NETCONF server does not return the namespace associated with an optional YANG model, the NETCONF client determines that the O-RU does not support the optional capability associated with the model.

In addition, for each supported schema, the ietf-yang-library lists the YANG feature names from this module that are supported by the server. The details of optional models and features are defined in Annex C.

6.3 M-Plane Operational State

The o-ran-mplane-int YANG model allows the O-RU to report the connectivity to NETCONF clients on a per sub-interface level. The client information includes the IP address(es) for the client(s) as well as the link-layer address used to forward packets towards the various management plane clients.

Chapter 7 Performance Management

This chapter provides the description of scenarios related to performance management. It consists of 2 functions. One is for the measurement activation and the other is the collection of measurement results.

7.1 Measurement Activation and De-activation

The measurement activation at the start-up installation is also allowed as described in chapter 3.

Pre-condition:

M-Plane is operational.

Post-condition:

Measurement is activated or deactivated as per NETCONF client's request.

This sub-section provides information about how to activate and de-activate the performance measurement via NETCONF <**edit-config**> to O-RU. The performance measurement is defined as o-ran-performance-management YANG module. In case of multiple NETCONF clients, only one NETCONF client shall activate/deactivate the measurements in the O-RU.

In the performance-management YANG module, the following parameters are defined.

- group of the measurement results, e.g., **transceiver-measurement-result** and, **rx-window-measurement-object**.
- **measurement-interval**: measurement interval for the **measurement-object**s to measure the performance periodically, e.g., 300, 600, 900 seconds. It is defined per the group of the measurement result.
- **measurement-object**: target metric to measure the performance, e.g., RX_POWER, TX_POWER, defined as key parameter.
- **active**: enable/disable the performance measurement per **measurement-object.** This value is Boolean. Default is FALSE.
- **start-time** and **end-time**: to report the time of measurement start and end for the **measurement-object** at each **measurement-interval**.
- **object-unit**: unit to measure the performance per object, e.g., O-RU, physical port number, antenna, carrier. The **object-unit** may be configurable Identifier **object-unit-id** means e.g., physical port number when object unit set to physical port number.
- **report-info**: the reporting info to the **measurement-object**, e.g., MAXIMUM, MINIMUM, FIRST, LATEST, FREQUENCY_TABLE and COUNT. Multiple info can be considered for one object if necessary.
- Optional configurable parameter(s) for **report-info**: some configurable parameters to report, e.g., **function**, **bin-count**, **upper-bound**, **lower-bound**. For the **bin-count** configuration, it shall be less than the parameter **max-bin-count** that is the capability information of NETCONF server for the maximum configurable value for bin-count.
- Additional reporting information for **report-info**: some additional information to report info, e.g., date-and-time.

The detail of the parameters per **measurement-object** and the group of measurement result are defined in Annex B.

The **measurement-interval** of **measurement-object** may be set to common or different values per group of the measurement result.

It is allowed that the measurement is activated and deactivated at any time. When different parameter measurements have intervals with a common factor, the O-RU shall synchronize the boundary of these measurements aligned with this factor, irrespective of when the different measurements are activated. And all of start points of the **measurement-interval**s shall be synchronized to zero o'clock mid-night by using an equation {full seconds (hour, minute and second) modulo '**measurement-interval**' = 0}, in order to ensure the same start and end of the **measurement-interval**s between O-RUs. For more detail see the following illustration.

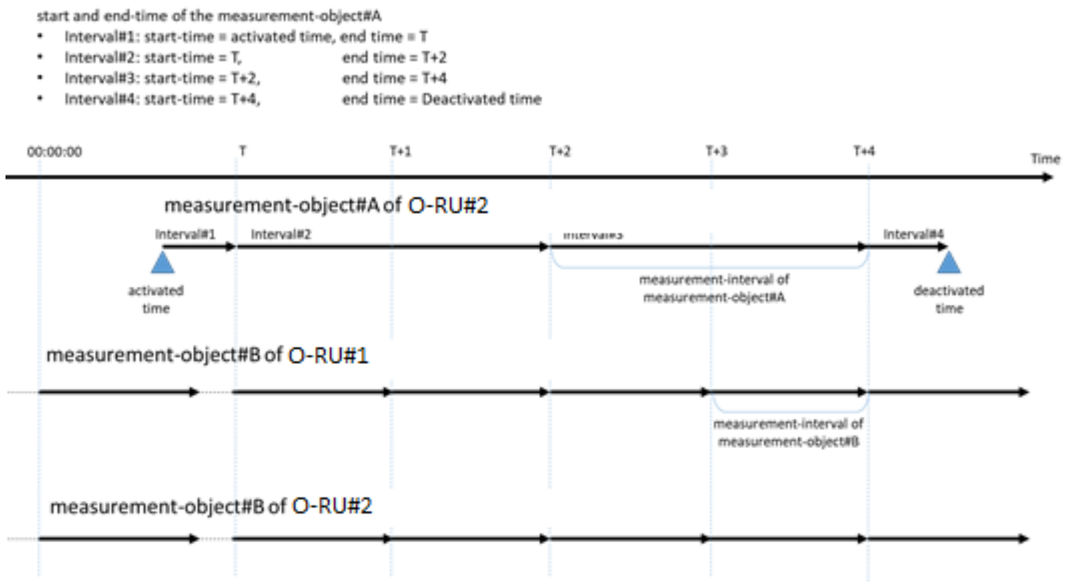

start and end-time of the measurement-object#A
• Interval#1: start-time = activated time, end time = T
• Interval#2: start-time = T, end time = T+2
• Interval#3: start-time = T+2, end time = T+4
• Interval#4: start-time = T+4, end time = Deactivated time
00:00:00
T
T+1
T+2
T+3
T+4
Time
measurement-object#A of O-RU#2
Interval#1
Interval#2
Interval#3
Interval#4
activated time
measurement-interval of measurement-object#A
deactivated time
measurement-object#B of O-RU#1
measurement-interval of measurement-object#B
measurement-object#B of O-RU#2

Figure 25: synchronization of measurement-interval.

The modification of the configurable parameters for the measurement shall be allowed while **active** for the corresponding **measurement-object** is set to FALSE.

All of the measurements are optionally supported in the O-RU.

The **report-info**, e.g., **count**, shall be started from 0 at the boundary of every **measurement-interval**. No accumulation is applied between the **measurement-interval**s.

7.2 Collection and Reporting of Measurement Result

This sub-section provides the description of scenarios used to collect measurement results. There are two options.

- NETCONF process: Create-subscription from NETCONF client and NETCONF notification from NETCONF server are used.

- File Management process: File upload mechanism is used for the measurement file from O-RU to configured file server(s) that O-RU can reach to.

Both methods are mandatory for the O-RU. The method(s) to be used is the matter of NETCONF client.

In case of multiple NETCONF clients, the O-RU shall report the same notification-based measurement results to all subscribed NETCONF clients, and the O-RU shall upload file-based results to all configured fileservers.

7.2.1 NETCONF process

This process needs the NETCONF capability: urn:ietf:params:netconf:capability:notification:1.0

1. NETCONF client subscribes to one or more measurement group(s) and/or **measurement-object**(s) to collect the measurement result by sending NETCONF <**subscribe-notification**> to NETCONF server in the O-RU. In this message, startTime and stopTime for the notification may be configurable. NETCONF client can configure the **notification-interval** in the performance-measurement YANG module.

2. NETCONF server sends NETCONF notification messages periodically to the client as configured by the **notification-interval**. The NETCONF notification message contains subscribed measurement group(s) and/or **measurement-object**(s). The **notification-interval** doesn't need to be same as the **measurement-interval**. The notification timing different from the **measurement-interval** is a matter to O-RU implementation.

This procedure is described in the following figure.

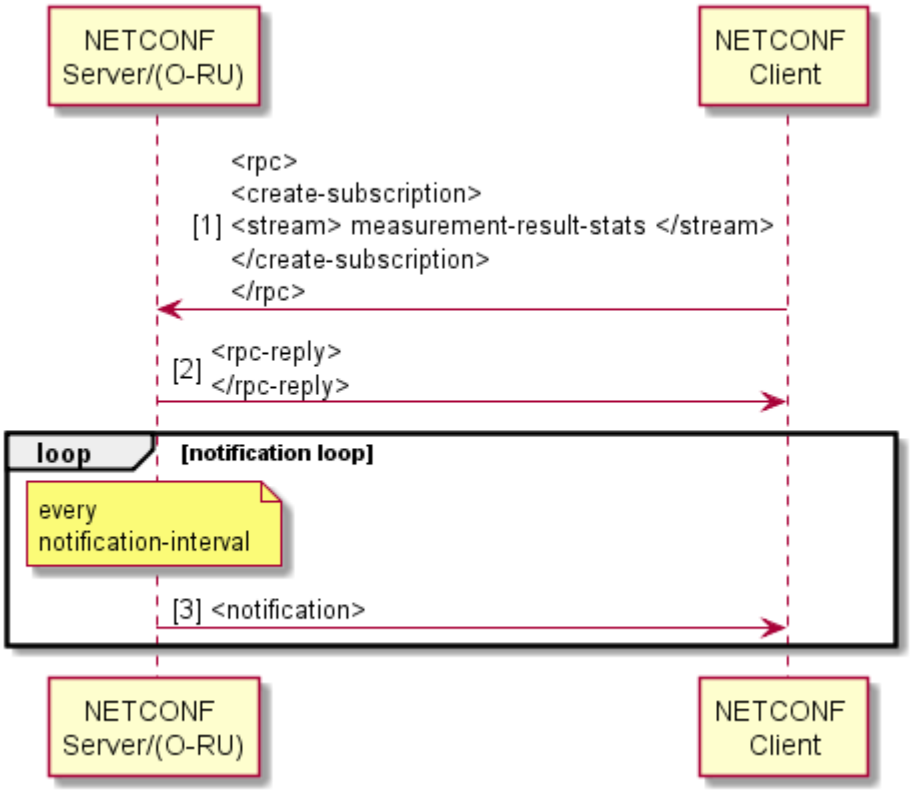


Figure 26: NETCONF process of Measurement Result Collection

Note: This figure uses **create-subscription** for the single stream "**measurement-result-stats**". In order to subscribe multiple notifications, the appropriate **create-subscription** message is required. Please refer to setion 8.2 for the appropriate example of **create-subscription** of multiple notifications.

In order to terminate the subscription, the NETCONF client shall send <**close-session**> operation from the subscription session. If NETCONF session is terminated by <**kill-session**>, the subscribed notification is terminated as well.

This procedure is described in the following figure.

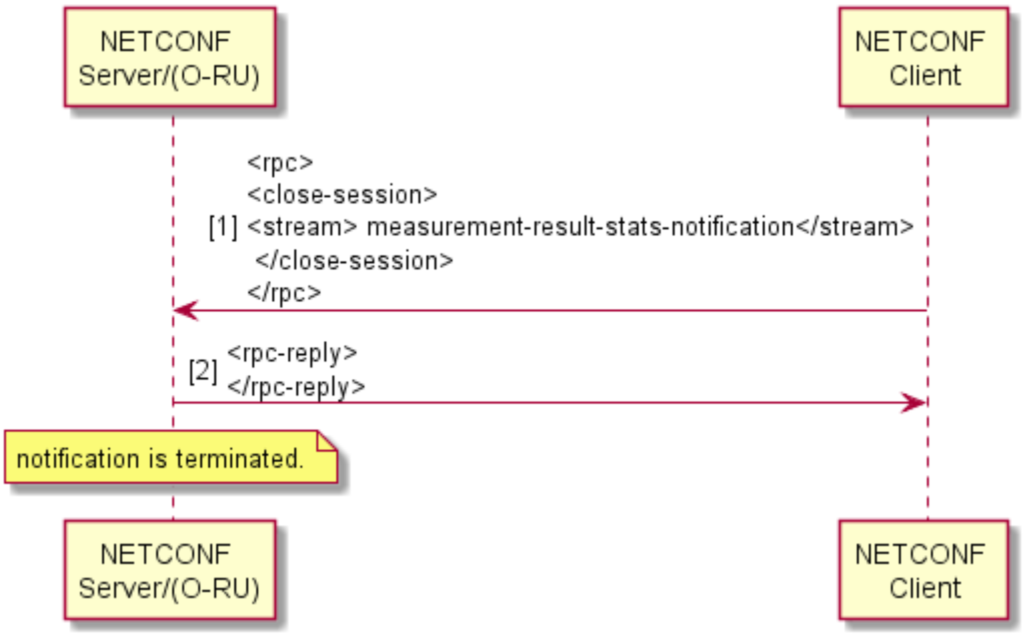


Figure 27: NETCONF process of Measurement Result Collection to end

When **notification-interval** is larger than **measurement-interval**, one notification may contain multiple stats which have consecutive periods indicating **start-time** and **end-time** for the measurement.

When **notification-interval** is smaller than **measurement-interval**, one notification may not contain the stats which **start-time** and **end-time** are not applicable to the period for the notification.

For example, when **notification-interval** = 60min, **measurement-interval** for **measurement-object**#A= 30min and **measurement-interval** for **measurement-object**#B = 15min, one notification contains 2 measurement results for **measurement-object**#A with consecutive **start-time** and **end-time**, and 4 measurement results for **measurement-object**#B with consecutive **start-time** and **end-time**.

For the other example, when **notification-interval** = 15min, **measurement-interval** for **measurement-object**#A= 30min and **measurement-interval** for **measurement-object**#B = 15min, one notification contains one measurement results for **measurement-object**#B but not for **measurement-object**#A. next notification contains both measurements result for **measurement-object**#A and #B.

7.2.2 File Management process

NETCONF client needs to configure a parameter of performance measurement YANG module '**enable-SFTP-upload**' to enable or disable the periodic file upload mechanism via NETCONF <**edit-config**>. Its default is FALSE.

In addition, the performance measurement YANG module defines **file-upload-interval**, **remote-SFTP-upload-path**, **credentials** information of the file server and **enable-random-file-upload** as configurable parameters.

When the parameter **enable-SFTP-upload** is set to TRUE, O-RU shall store the performance measurement files in the generic folder in O-RU, i.e., O-RAN/PM/. Every **file-upload-interval**, O-RU pushes the latest file to upload to the **remote-SFTP-upload-path** of the configured SFTP servers if **enable-SFTP-upload** is set to TRUE. Otherwise, the performance measurement file is not created and uploaded. The number of maximum performance files to be stored in O-RU simultaneously is a matter for O-RU implementation. The O-RU shall manage its own storage space by deleting the older files autonomously.

The O-RU shall ensure that the **start-time** and the **end-time** within the name of the performance measurement file are synchronized with the same manner as **measurement-interval** by using **file-upload-interval**.

If the parameter **enable-random-file-upload** is set to TRUE, the O-RU shall randomize the timing to upload SFTP file after the performance measurement file is ready to upload. The randomized timing is an O-RU implementation matter and shall not be later than next **file-upload-interval**.

The file name of the performance measurement is:

C<**start-time**>_<**end-time**>_<**name**>.csv

- Starting with a capital letter "C".
- Format of <**start-time**> and <**end-time**> can be local time or UTC.

  Local time format is YYYYMMDDHHMM+HHMM, indicating, year, month, day, hour, minute, timezone "+" or "-", hour and minute for the time zone.

  UTC format is YYYYMMDDHHMMZ, indicating, year, month, day, hour, minute and with a special UTC designator ("Z")

  Time zone offset is provided by **timezone-utc-offset** in o-ran-operation.yang.
- <**name**> in ietf-hardware is used
- "_" underscore is located between <**start-time**>, <**end-time**> and <**name**>
- File extension is "csv" as csv format file.

Example of measurement file is:

C201805181300+0900_201805181330+0900_ABC0123456.csv.

The file format of the performance measurement has following rule:

1. Each line starts with the **measurement-object** identifier, which measurement can be switched to TRUE or FALSE by **active** parameter. The identifier of each **measurement-object** is defined in Annex B.
2. After the **measurement-object** identifier, the name of **measurement-object**, **start-time**, **end-time** are followed.
3. Since the **report-info** results of any **measurement-object** are measured per **object-unit**, **object-unit-id** and set of **report-info** are repeated in one line.
4. When multiple **report-info** parameters exist per **object-unit**, all of the **report-info** are consecutively listed until the next **object-unit-id**. The order of parameters, such as object-unit-id, report-info and additional information for the report-info, shall be same as the order of those listed in NETCONF notification defined in o-ran-performance-management YANG module.

Example of measurement result in one line is:

1, RX_ON_TIME, 2018-05-18T13:00:00+09:00, 2018-05-18T13:30:00+09:00, 0, 123, AAAA, 1, 123, BBBB, 2, 123, CCCC, 3, 123, DDDD

- Measurement-object-identifier: 1
- Name of **measurement-object**: RX_ON_TIME
- **start-time**: 2018-05-18T13:00:00+09:00 as measurement **start-time**.
- **end-time**: 2018-05-18T13:30:00+09:00 as measurement **end-time**
- EAXC_ID: 0
- Count for EAXC_ID#0 : 123
- **name** of **transport-flow** information: AAAA
- :
- EAXC_ID: 3
- Count for EAXC_ID#3 : 123
- **name** of **transport-flow** information: DDDD

When **file-upload-interval** is larger than **measurement-interval**, one performance measurement file may contain multiple lines for the stats which have consecutive periods indicating **start-time** and **end-time** for the measurement.

When **file-upload-interval** is smaller than **measurement-interval**, one performance measurement file may not contain the line for the stats which **start-time** and **end-time** are not applicable to the period for the performance measurement file.

For example, when **file-upload-interval** = 60min, **measurement-interval** for **measurement-object**#A= 30min and **measurement-interval** for **measurement-object**#B = 15min, one performance measurement file contains 2 measurement result lines for **measurement-object**#A with consecutive **start-time** and **end-time**, and 4 measurement result lines for **measurement-object**#B with consecutive **start-time** and **end-time** as followings:

1, RX_POWER, 2018-05-18T13:00:00+09:00, 2018-05-18T13:15:00+09:00, 0, 123

1, RX_POWER, 2018-05-18T13:15:00+09:00, 2018-05-18T13:30:00+09:00, 0, 123

1, RX_ON_TIME, 2018-05-18T13:00:00+09:00, 2018-05-18T13:30:00+09:00, 0, 123, AAAA, 1, 123, BBBB, 2, 123, CCCC, 3, 123, DDDD

1, RX_POWER, 2018-05-18T13:30:00+09:00, 2018-05-18T13:45:00+09:00, 0, 123

1, RX_POWER, 2018-05-18T13:45:00+09:00, 2018-05-18T14:00:00+09:00, 0, 123

1, RX_ON_TIME, 2018-05-18T13:30:00+09:00, 2018-05-18T14:00:00+09:00, 0, 123, AAAA, 1, 123, BBBB, 2, 123, CCCC, 3, 123, DDDD

For the other example, when **file-upload-interval** = 15min, **measurement-interval** for **measurement-object**#A= 30min and **measurement-interval** for **measurement-object**#B = 15min, one performance measurement file contains one measurement result line for **measurement-object**#B but not for **measurement-object**#A. next performance measurement file contains both measurements result for **measurement-object**#A and #B as follows.

C201805181300Z+0900_201805181315+0900_ABC0123456.csv.

1, RX_POWER, 2018-05-18T13:00:00+09:00, 2018-05-18T13:15:00+09:00, 0, 123

C201805181315Z+0900_201805181330+0900_ABC0123456.csv.

1, RX_POWER, 2018-05-18T13:15:00+09:00, 2018-05-18T13:30:00+09:00, 0, 123

1, RX_ON_TIME, 2018-05-18T13:00:00+09:00, 2018-05-18T13:30:00+09:00, 0, 123, AAAA, 1, 123, BBBB, 2, 123, CCCC, 3, 123, DDDD

The performance measurement files stored in O-RAN/PM can be uploaded on-demand. For the file upload mechanism by on-demand way, **retrieve-file-list** and **file-upload** operations are used. For more detail, please refer to Chapter 9.

Chapter 8 Fault Management

Fault management is responsible for sending alarm notifications to the NETCONF Client. FM contains Fault Management Managed Element and via this Managed Element alarm notifications can be disabled or enabled.

The NETCONF Server is responsible for managing an "active-alarm-list". Alarms with severity "warning" are excluded from this list. When an alarm is detected it is added to the list; when the alarm reason disappears then the alarm is cleared - removed from the "active-alarm-list". Furthermore, when the element that was the "fault-source" of an alarm is deleted then all related alarms are removed from the "active-alarm-list".

The NETCONF Client can read "active-alarm-list" by **get** rpc operation.

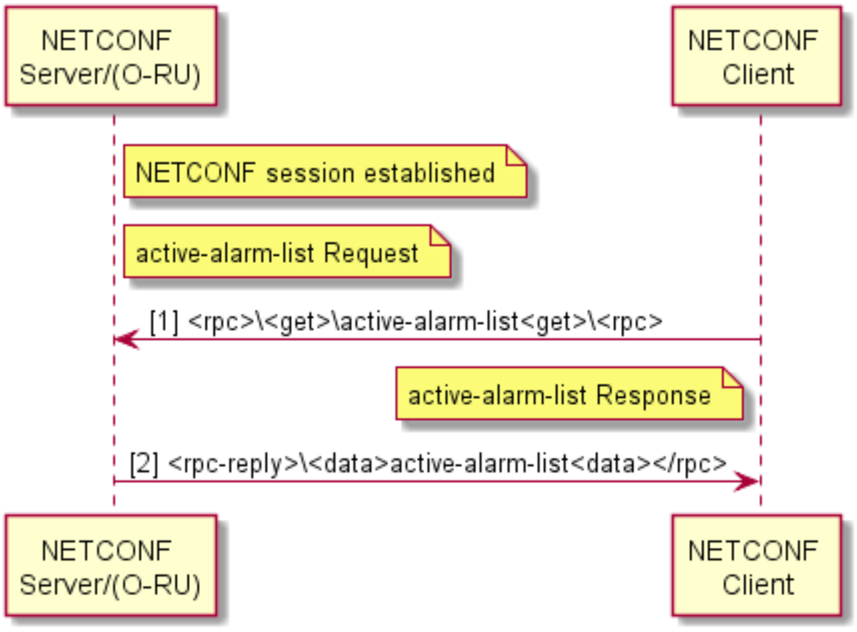

NETCONF Server/(O-RU)
NETCONF Client
NETCONF session established
active-alarm-list Request
[1] <rpc>\<get>\active-alarm-list<get>\<rpc>
active-alarm-list Response
[2] <rpc-reply>\<data>active-alarm-list<data></rpc>
NETCONF Server/(O-RU)
NETCONF Client

Figure 28: Read Active Alarms

8.1 Alarm Notification

The NETCONF Server is responsible to send <alarm-notif> to a NETCONF Client when the NETCONF Client has "subscribed" to alarm notification and:

- a new alarm is detected (this can be the same alarm as an already existing one, but reported against a different "fault-source" than the existing alarm)
- an alarm is removed from the list

Removal of alarms from the list due to deletion of "fault-source" element is considered as clearing and cause sending of <alarm-notif> to the NETCONF Client. This applies to alarms which were explicitly related to the deleted "fault-source" element. The rationale for such is to avoid misalignment between NETCONF Clients when one NETCONF Client deletes an element.

NETCONF Server reports in <alarm-notif> only for new active or cancelled alarms, not all active alarms.

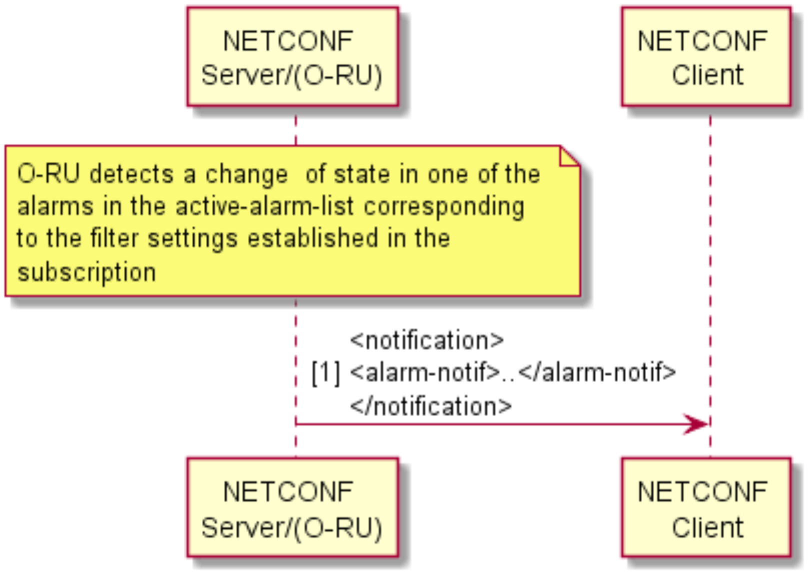

NETCONF Server/(O-RU)
NETCONF Client
O-RU detects a change of state in one of the alarms in the active-alarm-list corresponding to the filter settings established in the subscription
<notification>
[1] <alarm-notif>..</alarm-notif>
</notification>
NETCONF Server/(O-RU)
NETCONF Client

Figure 29: Alarm Notification

8.2 Manage Alarms Request

The NETCONF Client can “subscribe” to Fault Management Element by sending **create-subscription**, RFC5277 [21], to NETCONF Server.

RFC5277 allows <create-subscription> below:

```
<netconf:rpc netconf:message-id="101"
        xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <create-subscription
      xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <filter netconf:type="subtree">
      <event xmlns="http://example.com/event/1.0">
        <eventClass>fault</eventClass>
        <severity>critical</severity>
      </event>
      <event xmlns="http://example.com/event/1.0">
        <eventClass>fault</eventClass>
        <severity>major</severity>
      </event>
      <event xmlns="http://example.com/event/1.0">
        <eventClass>fault</eventClass>
        <severity>minor</severity>
      </event>
    </filter>
  </create-subscription>
</netconf:rpc>
```

Note: the NETCONF Client can disable/enable alarm sending only for all the alarms with same severity, not for single alarms.

The appropriate example for O-RAN YANG modules for **create-subscription** is as follows:

Case 1) NETCONF client subscribes **alarm-notif** filtering **fault-severity**: CRITICAL, MAJOR and MINOR and **measurement-result-stats** filtering **transceiver-stats** and **rx-window-stats** which **measurement-object** is RX_ON_TIME only:

```
<rpc xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <create-subscription
      xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <filter netconf:type="subtree">
      <alarm-notif xmlns="urn:o-ran:fm:1.0">
          <fault-severity>CRITICAL</fault-severity>
      </alarm-notif>
      <alarm-notif xmlns="urn:o-ran:fm:1.0">
          <fault-severity>MAJOR</fault-severity>
      </alarm-notif>
      <alarm-notif xmlns="urn:o-ran:fm:1.0">
          <fault-severity>MINOR</fault-severity>
      </alarm-notif>
      <measurement-result-stats xmlns="urn:o-ran:performance-management:1.0">
```

```
                <transceiver-stats/>
            </measurement-result-stats>
            <measurement-result-stats xmlns="urn:o-ran:performance-management:1.0">
                <rx-window-stats>
                                <measurement-object>RX_ON_TIME</measurement-object>
                            </rx-window-stats>
            </measurement-result-stats>
          </filter>
        </create-subscription>
      </rpc>
```

Case 2) NETCONF client subscribes default event stream NETCONF to receive all notifications defined in O-RAN YANG modules:

```
  <rpc xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
    <create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
          <stream>NETCONF</stream>
    </create-subscription>
  </rpc>
```

A high-level view of a NETCONF Client subscribing is shown directly below. After the NETCONF Client requests a subscription, the server sends an alarm-notif notification to the client when there is any change in the active alarms matching the filter specified in the subscription request.

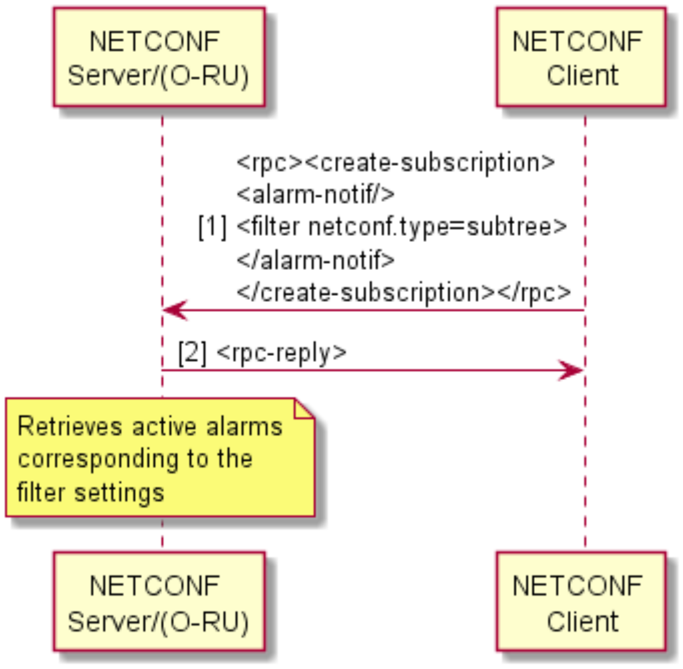

NETCONF Server/(O-RU)
NETCONF Client
[1] <rpc><create-subscription> <alarm-notif/> <filter netconf.type=subtree> </alarm-notif> </create-subscription></rpc>
[2] <rpc-reply>
Retrieves active alarms corresponding to the filter settings
NETCONF Server/(O-RU)
NETCONF Client

Figure 30: Manage Alarms Subscription Request

To terminate the subscription, the NETCONF client shall send a <close-session> operation from the subscription's session.

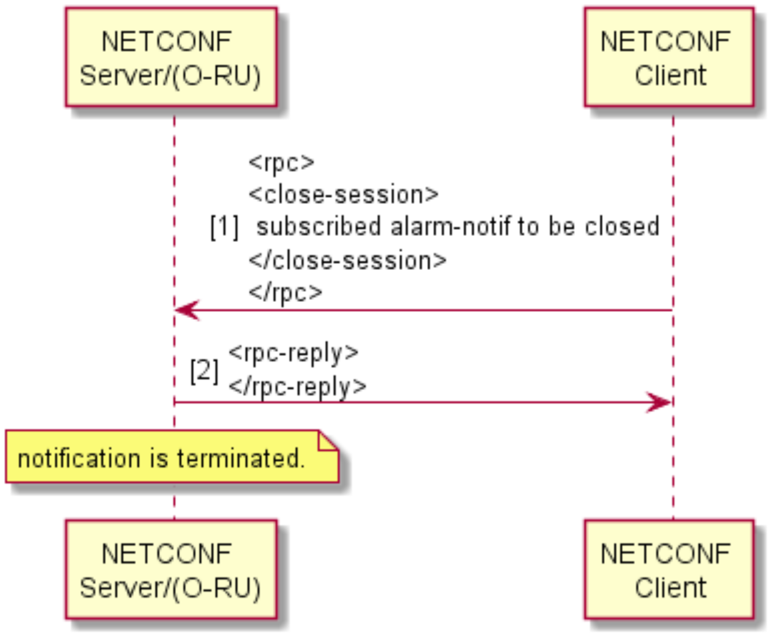


Figure 31: Terminating an Alarm Subscription

8.3 Fault Sources

Alarm notifications reported by NETCONF Server contain element “fault-source” which indicates the origin of an alarm. In general values of “fault-source” are based on names defined as YANG leafs:

- Source (Examples: fan, module, PA, port)

indicates that origin of the alarm within the O-RU. Value of “fault-source” is based on element name.

> Note: In case the NETCONF Server reports an unknown “fault-source”, the NETCONF Client can discard the <alarm-notif>.

- Source (other than when an element is within the O-RU)

Value of fault-source may be empty or may identify the most likely external candidate; for example, antenna line.

Alarms with different “fault-id”, “fault-source” or "fault-severity" are independent:

- Multiple alarms with same “fault-id” may be reported with different “fault-source”.
- Multiple alarms with same “fault-source” may be reported with different “fault-id”.
- When an alarm with a "fault-id" and a "fault-source" is reported with a "fault-severity" and its severity of alarm condition is upgraded or degraded, NETCONF server reports a new alarm with the same "fault-id" and the same "fault-source" with the updgraded or degraded "fault-severity" with "is-cleared"::FALSE and clears the previous alarm with the report of the "fault-id", "fault-source" and "fault-severity" with "is-cleard":: TRUE.

The range of "fault-id" is separated to common and vendor specific. The common fault-ids are defined in Annex A and more number will be used in future. The vendor specific range for the fault-id shall be [1000 .. 65535].

Alarm notifications reported by the NETCONF Server contain names of the “affected-objects” which indicate elements affected by the fault. In case the origin of the alarm is within the O-RU, other elements than “fault-source” which will not work correctly due to the alarm are reported via “affected-objects”. In case the origin of the fault is outside of the O-RU, the O-RU elements which will not work correctly due to the fault are reported via “affected-objects”.

Chapter 9 File Management

This chapter specifies File Management for the O-RU. Following operations are supported as a File Management.

- upload (see subsection 9.2)

  File upload from O-RU to O-DU/NMS triggered by O-DU/NMS.

- retrieve file list (see subsection 9.3)

  O-DU/NMS retrieves the file list in O-RU.

- download (see subection 9.4)

  File download from O-DU/NMS to O-RU triggered by O-DU/NMS

  Note: file-download has different purpose with software-download specified in subsection 5.3. For example, file-download can be used for Beamforming configuration in subsection 12.4.

Following other sections are related with File Management.

- Subsection 7.2.2: File Management process (can be used for on demand file upload purpose, since subsection 7.2.2 covers periodic file upload)

- Subsection 11.2: Troubleshooting

- Subsection 12.4: Beamforming Configuration

9.1 File System Structure

The file System structure of the O-RU is represented as a logical structure that is used by the file management procedures defined in the rest of this chapter. If the O-RU's physical file structure differs from the logical file structure defined below, the O-RU is responsible for performing the mapping between the two structures.

The O-RU shall support the standardized logical folders. In this version of the M-Plane specification, the following standardized folders are defined:

O-RAN/log/

O-RAN/PM/

O-RAN/transceiver/

And for those O-RU's supporting beamforming

O-RAN/beamforming/

The O-RU may additionally support vendor defined folders which are out of scope of this specification.

9.2 File Management Operation: upload

This subsection describes file upload method from O-RU to O-DU/NMS. sFTP is used for File management, and one file can be uploaded by one upload operation. The O-DU/NMS triggers file upload operation to O-RU.

Simultaneous multiple file upload operations can be supported under the same sFTP connection between O-RU to O-DU/NMS. If the O-RU has the number of limitation to upload simultaneously as a capability, it is allowed that O-RU reports failure notification for the upload request which is larger than the capability. The behaviour of O-DU/NMS is out of scope when O-DU/NMS receives failure notification from the O-RU.

Following rpc is used for upload operation.

-rpc: **file-upload**

- input

  - local-logical-file-path: the logical path of file to be uploaded (no wildcard is allowed)
  - remote-file-path: URI of file on the O-DU/NMS
- output
  - status: whether O-RU accepted or rejected the upload request
  - reject-reason: the human readable reason why O-RU rejects the request (only applicable if status is rejected)

In the rpc-reply, status whether the O-RU receives the upload request or rejects due to some reason (e.g., the number of limitation to upload simultaneously) is replied. If rejected, the human readable reject reason is also replied.

In notification, the result of the upload process (successfully uploaded or failed upload) is replied in addition to local-logical-file-path and remote-file-path. If failure, the human readable reason is also replied.

Figure 32 shows the file upload sequence diagram.

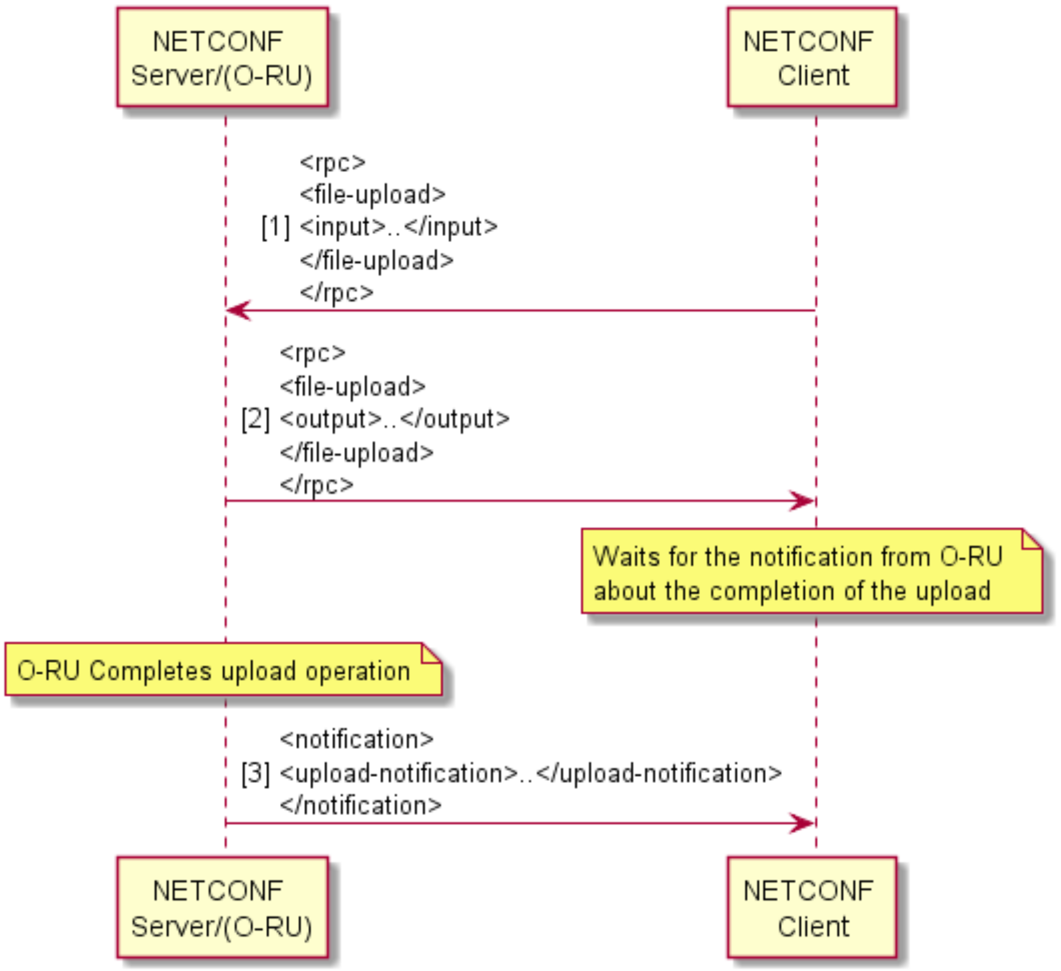


Figure 32: File Upload Sequence

9.3 File Management Operation: retrieve file list

This subsection describes file retrieve method which the O-DU/NMS retrieves the file list from the O-RU. One or multiple files' information can be retrieved by one retrieve file list operation (use of wildcard is allowed). The O-DU/NMS triggers the retrieve file list operation from the O-RU.

The following rpc is used for retrieve file list operation.

-rpc: **retrieve-file-list**

- input
    - logical path: the logical path of files to be retrieved as specified below (* is allowed as wild-card)
    - file-name-file name filter: the files which has the “file name filter” in the file name (* is allowed as wild-card)
- output
    - status: whether O-RU accepted or rejected the retrieve file list request
    - reject-reason: the human readable reason why O-RU rejects the request (only applicable if status is rejected)
    - file list

In rpc-reply, status whether the O-RU accepts the retrieve-file-list request or rejects due to some reason is replied. If rejected, the human readable reject reason is also replied.

Figure 33 shows the retrieve file list sequence diagram.

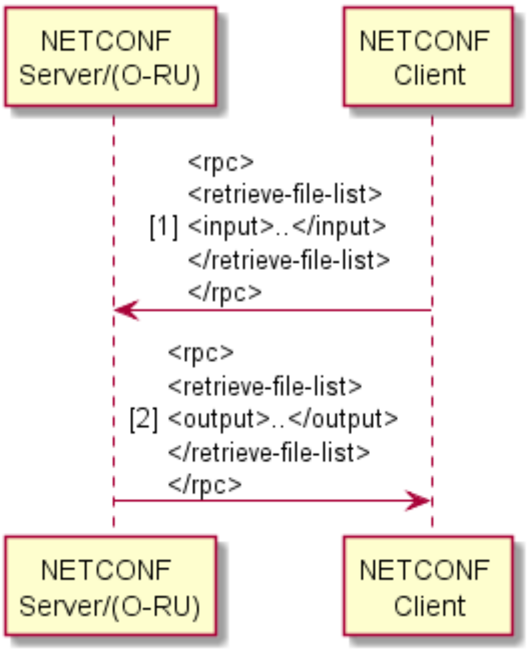

NETCONF Server/(O-RU)
NETCONF Client
[1] <rpc> <retrieve-file-list> <input>..</input> </retrieve-file-list> </rpc>
[2] <rpc> <retrieve-file-list> <output>..</output> </retrieve-file-list> </rpc>
NETCONF Server/(O-RU)
NETCONF Client

Figure 33: Retrieve File List Sequence

9.4 File Management Operation: download

This sub-chapter describes the file download method from O-DU/NMS to O-RU. sFTP is used for File management, and one file can be downloaded by one download operation. The O-DU/NMS triggers the file download operation to O-RU.

Simultaneous multiple file download operations can be supported under the same sFTP connection between the O-RU and O-DU/NMS. If the O-RU has the number of limitation to download simultaneously as a capability, it is allowed that the O-RU reports a failure notification for the download request which is larger than the capability. The behaviour of the O-DU/NMS is out of scope when O-DU/NMS receives failure notification from O-RU.

The following rpc is used for download operation.

-rpc: **file-download**

- input
    - local-logical-file-path: the logical path of file to be downloaded (no wildcard is allowed)
    - remote-file-path: URI of file on the O-DU/NMS
- output

- status: whether O-RU accepted or rejected the download request
- reject-reason: the human readable reason why O-RU rejects the request (only applicable if status is rejected)

In rpc-reply, status whether the O-RU receives the download request or rejects due to some reason (e.g., the number of limitation to download simultaneously) is replied. If rejected, the human readable reject reason is also replied.

In notification, the result of the download process (successfully downloaded or download is failure) is replied in addition to the local-logical-file-path and remote-file-path. If failure, the human readable reason is also replied.

Figure 34 shows the file download sequence diagram.

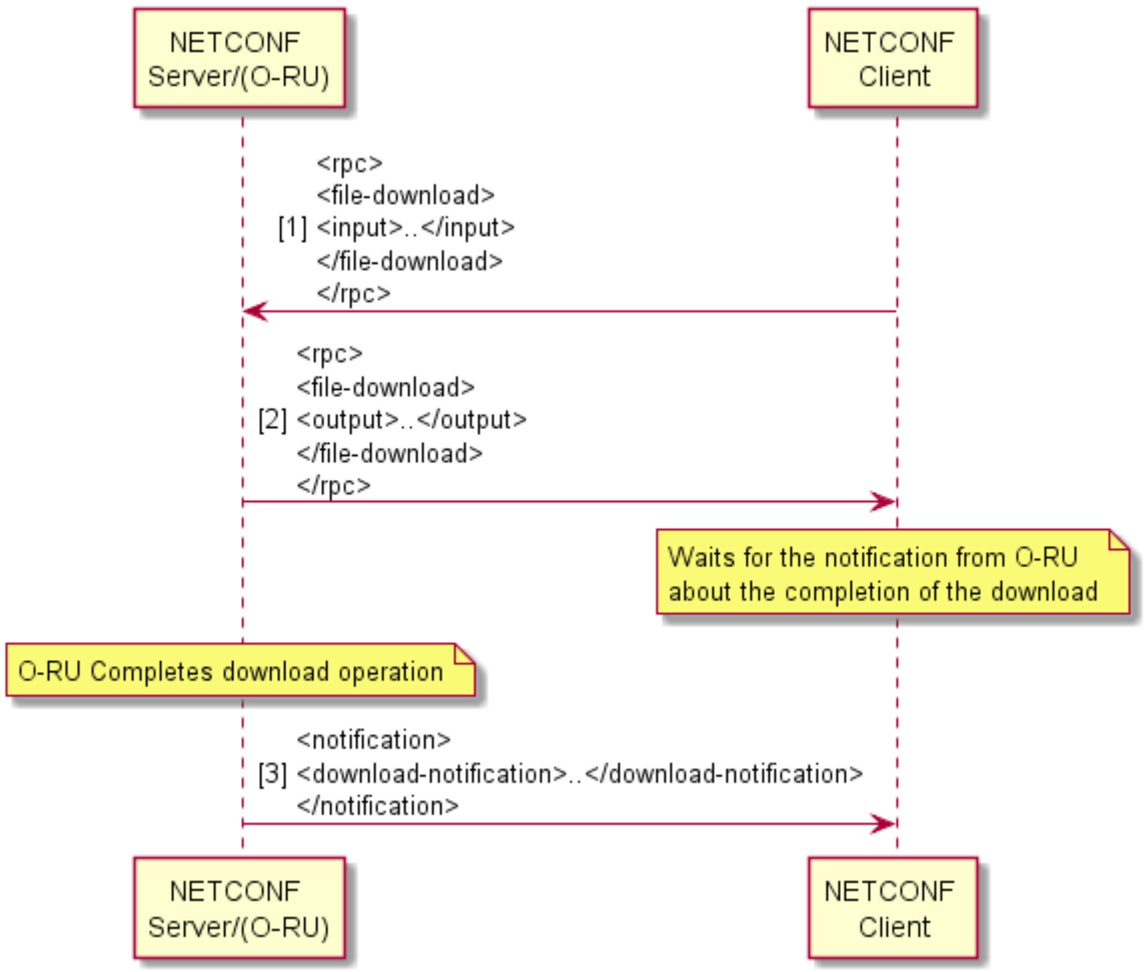


Figure 34: File Download Sequence

Chapter 10 Synchronization Aspects

This chapter provides the Management Plane's interactions with various aspects of the time synchronization of the O-RU. In general, the O-RU is responsible for managing its synchronization status, to select one or more synchronization input source(s) (based on vendor specific implementation) and assure that the resulting accuracy meets that required by the Radio Access Technology being implemented.

10.1 Sync Status Object

This **sync** container provides synchronization state of the module. If the O-DU is interested in Sync status, it may send NETCONF <subscribe-notification> to the NETCONF Server in the O-RU. Event notifications will be sent whenever the state of the O-RU synchronization changes.

The State of O-RU synchronization is indicated by the following allowed values:

- **LOCKED:** O-RU is in the locked mode, as defined in ITU-T G.810.
- **HOLDOVER:** O-RU clock is in holdover mode.
- **FREERUN:** O-RU clock isn't locked to an input reference and is not in the holdover mode.

The **sync** container allows the O-RU to list via an array the synchronization sources which it is capable of supporting. The allowed values are:

- GNSS
- PTP
- SYNCE

10.2 Sync Capability Object

The module's synchronization capability is provided via this object. This indicates the accuracy of the derived Telecom Slave Clock (T-TSC) to which the module's design is capable of. For details on the actual capability levels, see section 9.3 of the O-RAN WG4 CUS plane specification [2]. There are two enumerations possible:

- CLASS_B
- ENHANCED

10.3 PTP Configuration

This container defines the configuration of Precision Time Protocol.

domain-number

This parameter indicates the Domain Number for PTP announce messages. Allowed values: 0 ~ 255.

Default: 24.

Note: ITU-T G.8275.1 [22] uses domain numbers in the range 24...43, but the entire range is allowed to ensure flexibility of the M-Plane specification. For ITU-T G.8275.2 domain numbers from range 44…63 shall be used.

accepted-clock-classes

Contains the list of PTP acceptable Clock Classes, sorted in the descending order.

Note: The sender must generate the list of acceptable clock classes. The list must be sorted in descending order. Each accepted Clock Class value must appear only once in the list. Depending on implementation, the receiver may interpret the list in either of two ways:

a) use only the first (i.e. the maximum) item in the list, interpreting it as a threshold value for acceptable clock classes, while ignoring all other items in the list;

b) use the whole list, interpreting it as an explicit list of acceptable clock classes.

Default: 7, 6

clock-class

The PTP Clock Class accepted by the O-RU. Allowed values: 0 ~ 255.

Note: Not all values are compliant to [22], but the entire range is allowed in M-plane specification to ensure flexibility. The values can be validated/filtered on the receiver side, if necessary.

ptp-profile

Defines which PTP profile will be used.

Allowed values:

- G_8275_1 (multicast over Ethernet will be used, see: ITU-T G.8275.1)
- G_8275_2 (unicast over IP will be used, see: ITU-T G.8275.2)

Default: G_8275_1.

delay-asymmetry

Defines the static phase error in the recovered PTP timing signal to be compensated at the O-DU. The error is defined in units of nanoseconds in the range ±10 000 ns. According to ITU-T G.810 [23] and IEEE1588v2-2008 [24] the sign of the parameter is interpreted as follows:

- If the phase error to be compensated is negative, then the recovered timing signal shall be advanced by the time interval equal to the configured value to compensate the error.
- If the phase error to be compensated is positive, then the recovered timing signal shall be delayed by the time interval equal to the configured value to compensate the error.

Default: 0

Note: Modification of this parameter may have impact on RF transmission, but shall occur without unit restart.

Note: This parameter is optional for support. If the O-RU does not support this value, the O-RU uses the default value. If the O-RU does not support manual compensation, it ignores the parameter setting.

Note: Granularity of the applied value depends on the architecture and implementation of the system clock, and therefore, may vary across vendors.

10.3.1 G.8275.1 specific parameters

multicast-mac -address

The parameter defines the destination MAC address, used by the O-RU in the egress PTP messages.

Allowed values:

- FORWARDABLE (means that PTP shall use 01-1B-19-00-00-00 destination MAC address)
- NONFORWARDABLE (means that PTP shall use 01-80-C2-00-00-0E destination MAC address)

Default value: FORWARDABLE.

10.3.2 G.8275.2 specific parameters

This section contains G.8275.2 specific parameters

local-ip-port

The parameter defines local ip address which will be used as a slave port for receiving ptp signal

master-ip-configuration

The parameter defines list of ip configuration of devices acting as ptp signal source.

local-priority

The parameter defines local priority or underlying master IP address.

ip-address

the parameter defines master IP address.

log-inter-sync-period

The parameter defines number of sync message during 1 second

Allowed values: 0 ~ -7 (this represents the value from 1 message per second to 128 messages per second)

log-inter-announce-period

The parameter defines number of announce message during 1 second

Allowed values: 0 ~ -3 (this represents the value from 1 message per second to 8 messages per second)

10.4 PTP Status

The PTP Status container is used to collect operational status information of the PTP ordinary clock, controlled by the O-RU. The object may be used to display operational information, which facilitates troubleshooting, to the operator. The information in the object shall not be used by the O-DU to autonomously alter its operation. If the O-DU is interested in PTP status, it may NETCONF <subscribe-notification> to the NETCONF Server in the O-RU. Notifications will only indicate changes to the lock-state. Before requesting or subscribing to PTP status information, the O-DU shall ensure that PTP is supported by the O-RU by requesting the **supported-timing-reference-types**, as defined in this chapter (Section 10.1). The following list includes the related parameters of this container.

reporting-period

This parameter defines minimum period in seconds between reports, sent by the NETCONF Server, for parameters in this container.

default: 10

lock-state

This parameter indicates whether the integrated ordinary clock is synchronizing to the reference, recovered from PTP flow. The exact definition when to indicate locked or unlocked is up to specific implementation.

- **LOCKED:** The integrated ordinary clock is synchronizing to the reference, recovered from PTP flow.
- **UNLOCKED:** The integrated ordinary clock is not synchronizing to the reference, recovered from PTP flow.

clock-class

This parameter contains the clock class of the clock, controlled by the O-RU.

sources

This parameter contains characteristics of PTP sources of the clock, controlled by the O-RU.

state

This parameter indicates status of the PTP source:

- **PARENT:** Indicates that the PTP signal from this source is currently used as a synchronization reference.
- **OK:** Indicates that the PTP signal from this source can be potentially used as a synchronization reference, i.e. SSM messages, received from this source, contain acceptable clock quality level.
- **NOK:** Indicates that the PTP signal from this source cannot be used as a synchronization reference, i.e. SSM messages, received from this source, contain unacceptable clock quality level.
- **DISABLED:** Indicates that SSMs are not received from this SyncE source.

See the related o-ran-sync YANG Model for the full details.

10.5 SyncE Configuration

This container defines the configuration of SyncE

acceptance-list-of-ssm

The parameter contains the list of SyncE acceptable Synchronization Status Messages (SSM).

Allowed values:

- PRC (Primary Reference Clock)
- PRS (Primary Reference Source-Stratum 1)
- SSU_A (Synchronisation Supply Unit A)
- SSU_B (Synchronisation Supply Unit B)
- ST2 (Stratum 2)
- ST3 (Stratum 3)
- ST3E (Stratum 3E)
- EEC1 (Ethernet Equipment Clock 1)
- EEC2 (Ethernet Equipment Clock 2)
- DNU (Do Not Use)
- NONE

ssm-timeout

The parameter contains the value of maximum duration in seconds for which the actual SSM value may be different than configured values.

10.6 SyncE Status

The SyncE Status container is used to collect operational status information of SyncE reference on a node, controlled by O-RU. If the O-DU is interested in SyncE status, it may issue a NETCONF <subscribe-notification> to the NETCONF Server in the O-RU. Notifications will only indicate changes to the lock-state. Before requesting or subscribing to SyncE status information, the O-DU shall ensure that SyncE is supported at the O-RU by requesting the supported timing reference types, as defined earlier in this section 10.1. The following list summarizes the related parameters of this container.

reporting-period

This parameter defines minimum period in seconds between reports, sent by the NETCONF Server, for parameters in this container.

default: 10

lock-state

This parameter indicates, whether the integrated ordinary clock is synchronizing to the reference, recovered from the SyncE signal. The exact definition when to indicate locked or unlocked is up to specific implementation.

- **LOCKED:** The integrated ordinary clock is synchronizing to the reference, recovered from the SyncE signal.
- **UNLOCKED:** The integrated ordinary clock is not synchronizing to the reference, recovered from the SyncE signal.

sources

This parameter contains characteristics of SyncE sources of the clock, controlled by the NETCONF Server

state

This parameter indicates status of the SyncE source:

- **PARENT:** Indicates that the SyncE signal from this source is currently used as a synchronization reference.

- **OK:** Indicates that the SyncE signal from this source can be potentially used as a synchronization reference, i.e. SSM messages, received from this source, contain acceptable clock quality level.
- **NOK:** Indicates that the SyncE signal from this source cannot be used as a synchronization reference, i.e. SSM messages, received from this source, contain unacceptable clock quality level.
- **DISABLED:** Indicates that SSMs are not received from this SyncE source.

quality-level

This parameter contains value of the SSM clock quality level, received in SSM messages from the SyncE source.

See the related o-ran-sync YANG Model for the full details.

10.7 GNSS Configuration

This container defines the configuration of Global Navigation Satellite System (GNSS).

enable

This parameter defines if GNSS receiver shall be enabled or not. Allowed values: true/false;

Default values: false.

satellite-constelation-list

This parameter defines list of constellations to be used to acquire synchronization.

Allowed values:

- GPS
- GLONASS
- GALILEO
- BEIDOU

polarity

This parameter defines pulse polarity

Allowed values:

- POSITIVE
- NEGATIVE

Default value: POSITIVE.

cable-delay

This parameter is used to compensate cable delay. Allowe-values: 0 ~ 1000

Default value: 5

Note: This value is given in ns (nanoseconds) it is recommended to compensate 5ns per each meter of the cable.

anti-jam-enable {if feature GNSS-ANTI-JAM}

This parameter is used to enable or disable anti-jamming. Allowed values: true/false

Default value: false.

10.8 GNSS Status

An O-RU supporting GNSS capability uses the **gnss-state** container to report the state of its GNSS receiver. If the O-DU is interested in GNSS status, it may NETCONF <subscribe-notification> to the NETCONF Server in the O-RU before requesting or subscribing the GNSS status information. Notifications will only provide changes to the gnss-status. The O-DU shall ensure that GNSS is supported by the O-RU by requesting supported timing reference types, as defined in this chapter (Section 10.1). The following list summarizes the related parameters of this container.

gnss-status

This parameter indicates the status of the GNSS receiver:

- **SYNCHRONIZED:** Indicates that the GNSS receiver is synchronized.
- **ACQUIRING-SYNC**: Indicates the GNSS receiver is functioning correctly, but has not acquired synchronization
- **ANTENNA-DISCONNECTED:** Indicates the GNSS receiver is reporting that its antenna is disconnected.
- **INITIALIZING:** Indicates that the GNSS receiver is initializing.
- **ANTENNA-SHORT-CIRCUIT**: Indicates that the GNSS receiver is reporting that its antenna is short circuited.

Additionally, when the GNSS receiver is synchronized, the O-RU can report the following additional information:

satellites-tracked

The number of satellites being tracked by the O-RU receiver

altitude, **latitude** and **longitude**

The geospatial location reported by the GNSS receiver

Chapter 11 Operations Use Cases

11.1 Supervision Failure

Section 3.6 describes the procedure by which an O-RU uses the expiry of a supervision watchdog timer to trigger "supervision failure" condition.

Note, an O-RU operating in supervision failure may still have NETCONF sessions established with one or more NETCONF clients with non-"sudo" privileges.

An O-RU that detects a supervision failure shall immediately cease RF transmission. The O-RU shall perform an autonomous recovery reset procedure. This procedure will trigger the re-initialization of the transport layer and re-commencement of the start-up installation procedures defined in section 3.

11.2 Troubleshooting

By requesting technical logs an O-RU controller is able to get data files(s) that can be used for troubleshooting purposes.

The O-RU provides all possible files. The contents and log formats are dependent on O-RU implementation.

The number and size of files provided by O-RU is not restricted but the O-RU may keep the number and size of files reasonably small to allow completion of the whole "Troubleshooting data upload" scenario (all files) within 15 minutes (with target to complete within 3 minutes). It is also recommended to provide more useful files first.

Note, the O-RU controller is free to continue the scenario till completion past the allowed time or skip requesting further files.

The files should be compressed with compression method indicated by file name extension:

- .gz (DEFLATE),
- .lz4 (LZ4),
- .xz (LZMA2 - xz utils),
- .zip (DEFLATE - zlib library).

There are three operations for the troubleshooting, “start”, “stop” and “generate”.

<start-troubleshooting-logs> rpc triggers the O-RU to start collecting troubleshooting events and storing them in files.

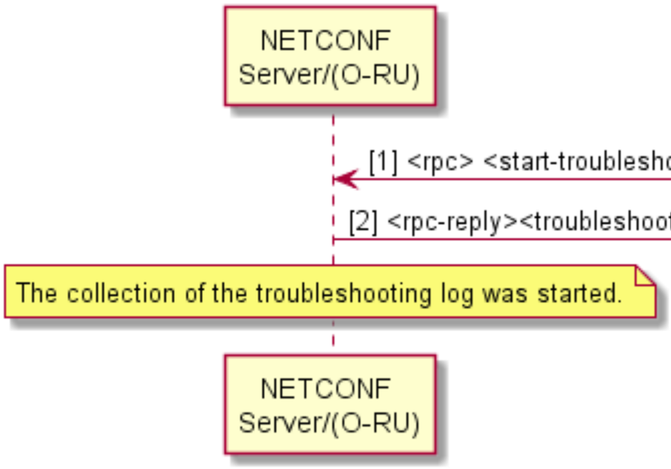

NETCONF Server/(O-RU)
NETCONF Client
[1] <rpc> <start-troubleshooting-logs>
[2] <rpc-reply><troubleshooting-status-grouping>
The collection of the troubleshooting log was started.
NETCONF Server/(O-RU)
NETCONF Client

Figure 35: Generating Troubleshooting Log

<stop-troubleshooting-logs> rpc triggers to the O-RU to stop collecting troubleshooting events and storing them in files.

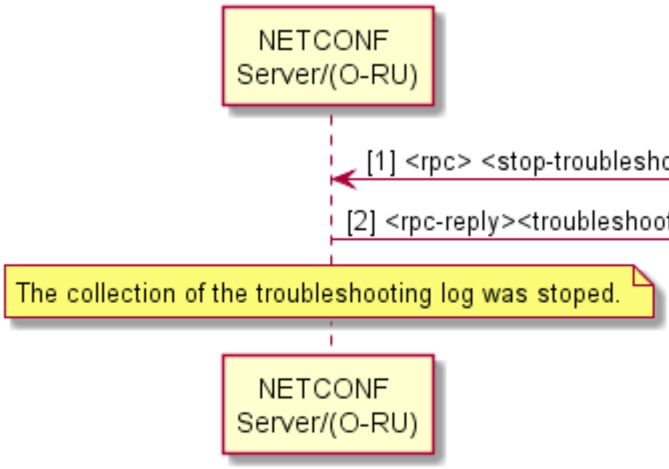

NETCONF Server/(O-RU)
NETCONF Client
[1] <rpc> <stop-troubleshooting-logs>
[2] <rpc-reply><troubleshooting-status-grouping>
The collection of the troubleshooting log was stoped.
NETCONF Server/(O-RU)
NETCONF Client

Figure 36: Stop Troubleshooting Log

<troubleshooting-logs-generated> notification notifies the NETCONF Client after the NETCONF Server has finished creating all troubleshooting logs, indicating to the NETCONF client that the logs are ready to be uploaded.

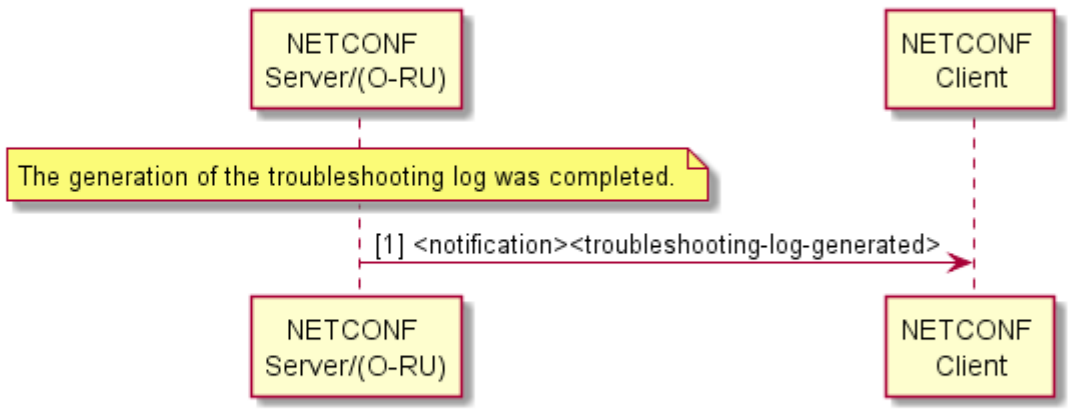

NETCONF Server/(O-RU)
NETCONF Client
The generation of the troubleshooting log was completed.
[1] <notification><troubleshooting-log-generated>
NETCONF Server/(O-RU)
NETCONF Client

Figure 37: Generate Troubleshooting Log

The time required to create logs depends on vendor design and may be after <start-troubleshooting-logs> or <stop-troubleshooting-logs>.

11.3 Operational aspects of Antenna Line Devices

An O-RU can connect to one or more external equipment such as a RET, MultiRET, MHA, RAE and etc.

For the communication with the external equipment, AISG 2.0 protocol [26] as Layer 7 application and HDLC protocol as Layer 2 data link are used.

- HDLC protocol is standardized by ISO/IEC 13239 (https://www.iso.org/standard/37010.html). Detailed information can also be found in TS 25.462 [27].

- AISG 2.0 protocol is standardized by "Control interface for antenna line devices Standard No. AISG v2.0" [26] which is an adaptation of Iuant interface application layer defined in TS 25.466 [28] and formerly in TS 25.463 [29].

An O-RU may provide one or more ALD ports supporting connection with Antenna Line Devices. Each ALD port shall be able to support more than one ALD (i.e., a chained ALD configuration).

This section describes the communication mechanisms based on AISG 2.0 protocol [26]. For communication with external equipment, AISG 2.0 uses Application Part protocols (RETAP, TMAAP etc.) at Layer 7 and HDLC as a Layer 2 datalink protocol.

Note, further definitions of O-RU support for Antenna Line Devices will be included in a future version of this specification, and will target support of the AISG 3.0 protocol.

11.3.1 HDLC Interworking

HDLC protocol is standardized by ISO/IEC 13239. Detailed information can also be found in TS 25.462 [27]. The AISG 2.0 protocol is standardized by "Control interface for antenna line devices Standard No. AISG v2.0" which is an adaptation of Iuant interface application layer defined in TS 25.466 [28] and formerly in TS 25.463 [29].

Note: the assumed HDLC communication speed is 9600 bits per second.

In order to handle collision detection in the HDLC branch, an O-RU supporting the ALD functionality shall support the following running counters reported using the corresponding YANG model:

- Frames with wrong FCS
- Frames without stop flag
- Number of received octets

For running counters served by the O-RU, both the O-RU and NETCONF Client shall handle wrap-over mechanism in a way, that wrap over zero is not considered as erroneous situation.

A NETCONF client can recover these counters. From the changes observed in above counters, a NETCONF Client can deduce the presence of a collision on the HDLC bus. Additional diagnostic information may be derived from how these counters are incrementing.

Additionally, the O-RU implements "RPC Status" to indicate status of last "ald-communication" RPC to requestor.

- Status - flow control indicator of last requested operation (Status of RPC).

Prior to any communication with ALD(s), the O-RU shall provide ALDs with DC power. The way of how DC power is managed is out of scope of this interface specification.

In order to support collision detection and flow control, the following reference architecture with functional split is defined:

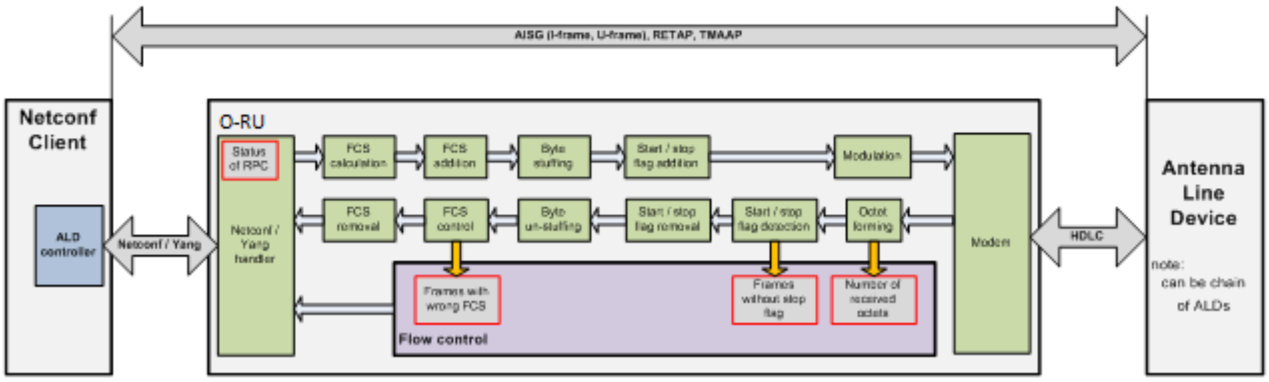


Figure 38: ALD Reference Architecture

The result of the above architecture is, that below mentioned parts of HDLC message are processed by entities as illustrated in Figure 39:

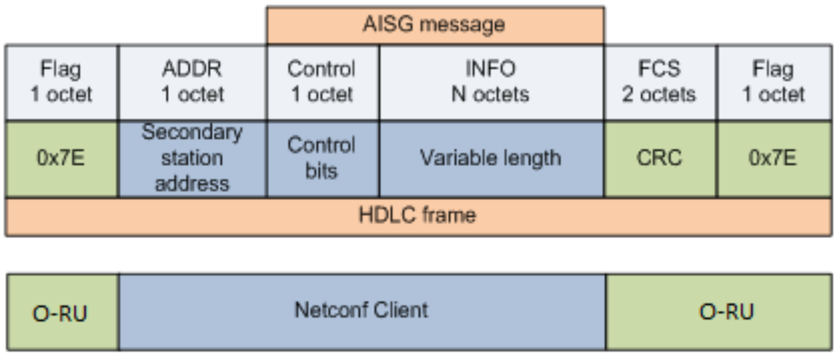


Figure 39: Component's responsibility split.

11.3.2 ALD Operations

The NETCONF Client sends RPC <ald-communication> to the O-RU. The RPC has following input parameters:

- leaf: **ald-port-id** (uint8) - contains the identity of the ALD port. The O-RU shall output the data to (corresponds to O-RU resources provided to NETCONF Client as inventory information)

- leaf: **ald-req-msg** (up to 1200 bytes) - may contain HDLC address, control bits and payload (see: TS 25.462 for details)

The O-RU performs HDLC communication with the ALD as follows: immediately after the requested payload is sent to the ALD over the desired ALD port, the O-RU switches the ALD port into reception mode.

Note, for details of HDLC transmission and reception algorithm, please see TS 25.462, chapter 4.5 "Message timing". Bits received within reception window are formed to octets and inserted as payload into ald-resp-msg.

The O-RU responds to the NETCONF Client using the <rpc-reply> message containing following parameters:

- leaf: **ald-port-id** (uint8)
- leaf: **status**
- leaf: **ald-resp-msg** (up to 1200 bytes)
- leaf: **frames-with-wrong-crc** (4 bytes)
- leaf: **frames-without-stop-flag** (4 bytes)
- leaf: **number-of-received-octets** (4 bytes)

Note: In case there is no response from the ALD received within the reception window, the record "**ald-resp-msg**" in <rpc-reply> sent by the O-RU shall be empty.

After reception, the O-RU shall wait an additional 3ms before the next transmission towards HDLC bus is initiated. See TS 25.462, chapter 4.5 "Message timing" for details.

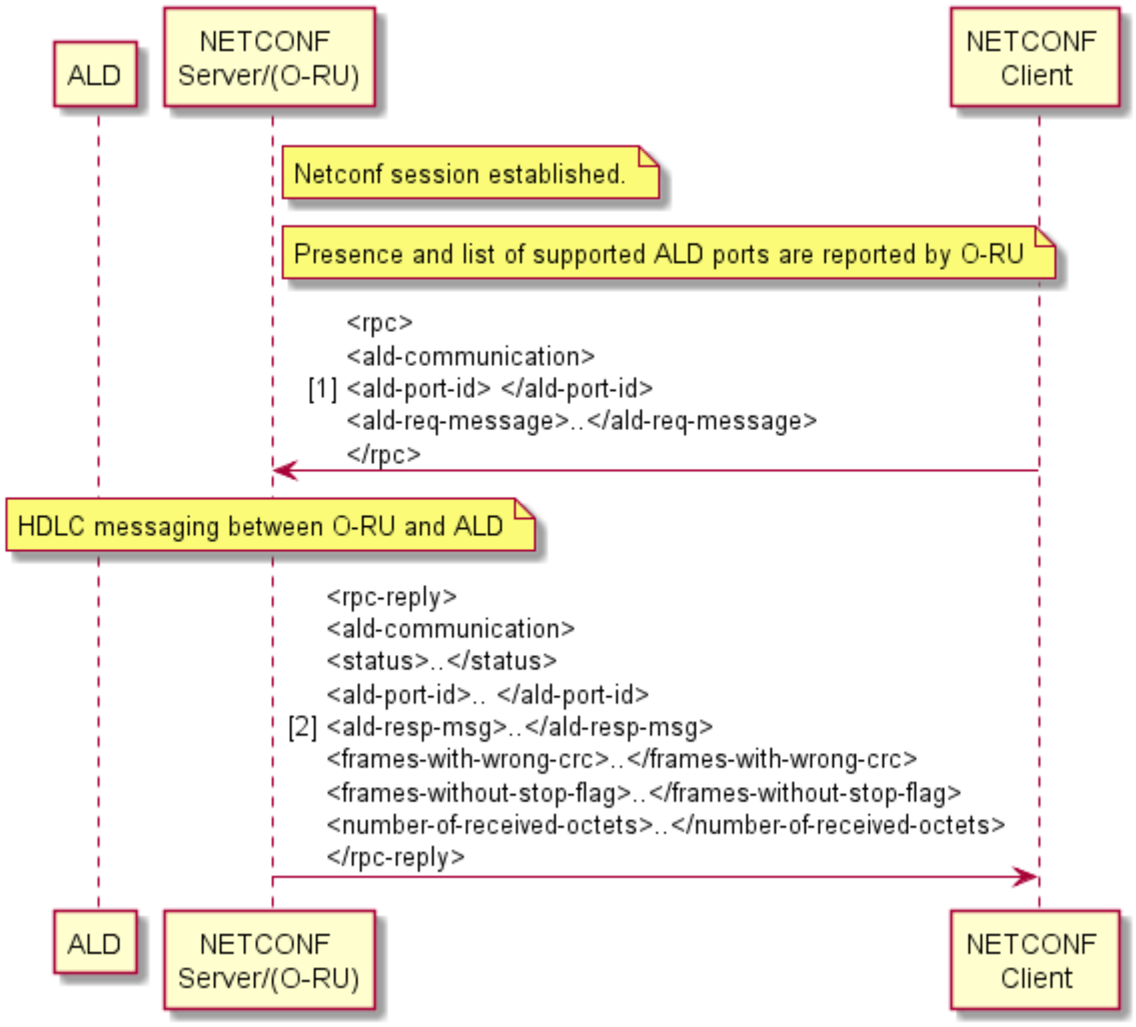

ALD
NETCONF Server/(O-RU)
NETCONF Client
Netconf session established.
Presence and list of supported ALD ports are reported by O-RU
[1]
<rpc>
<ald-communication>
<ald-port-id> </ald-port-id>
<ald-req-message>..</ald-req-message>
</rpc>
HDLC messaging between O-RU and ALD
[2]
<rpc-reply>
<ald-communication>
<status>..</status>
<ald-port-id>.. </ald-port-id>
<ald-resp-msg>..</ald-resp-msg>
<frames-with-wrong-crc>..</frames-with-wrong-crc>
<frames-without-stop-flag>..</frames-without-stop-flag>
<number-of-received-octets>..</number-of-received-octets>
</rpc-reply>
ALD
NETCONF Server/(O-RU)
NETCONF Client

Figure 40: ALD Message Transfer

General scenario

Precondition:

M-Plane connectivity between NETCONF Client and NETCONF Server is successfully established. NETCONF Server reports presence of the supported HDLC Primary Devices.

1) NETCONF Client triggers DC voltage on desired ALD ports using NETCONF <edit-config> RPC. After DC is turned on - NETCONF Client waits 3s.

2) NETCONF Client performs HDLC link speed alignment to assure that all ALDs connected to a particular port have switched themselves to the correct baud rate used by this port.

3) NETCONF Client performs HDLC bus scan using desired HDLC Primary Device offered by O-RU.

3) NETCONF Client determines presence of HDLC Secondary Devices.

4) NETCONF Client assigns HDLC addresses to desired HDLC Secondary Devices.

5) NETCONF Client initiates HDLC layer for secondary devices by sending SNRM command.

6) NETCONF Client starts polling procedure for every HDLC-addressed Secondary Device.

Postcondition:

Detected and addressed HDLC Secondary Devices are available for configuration.

11.4 Operational aspects of external IO

An O-RU can connect to one or more input and output ports for external device supervision and control.

The External IO has the following functions

- INPUT: Supervising external devices
- OUTPUT: Controlling external devices

Also external IO function include signalling to get the O-RU and O-RU controller in sync, enables port monitoring on the O-RU and provides notification from the O-RU to an O-RU controller, and provides control from an O-RU controller to the O-RU and enables output port controlling on the O-RU.

The O-RU only implements external IO yang module if the O-RU supports External IO aspect.

A change in condition of the external IO shall not affect other O-RU services such as RF transmission / reception behaviour.

11.4.1 External input

This section explains single external input line case. For multiple external inputs case, same behaviour for each input shall be processed individually.

For input, the O-RU and O-RU controller shall support two scenarios.

[1] To retrieve input state from O-RU controller and respond the input state from the O-RU to O-RU controller.

[2] To send notification from the O-RU to O-RU controller when input state is changed.

The value shall be

TRUE: Circuit is open.

FALSE: Circuit is closed

When nothing is connected to the line the value shall be TRUE.

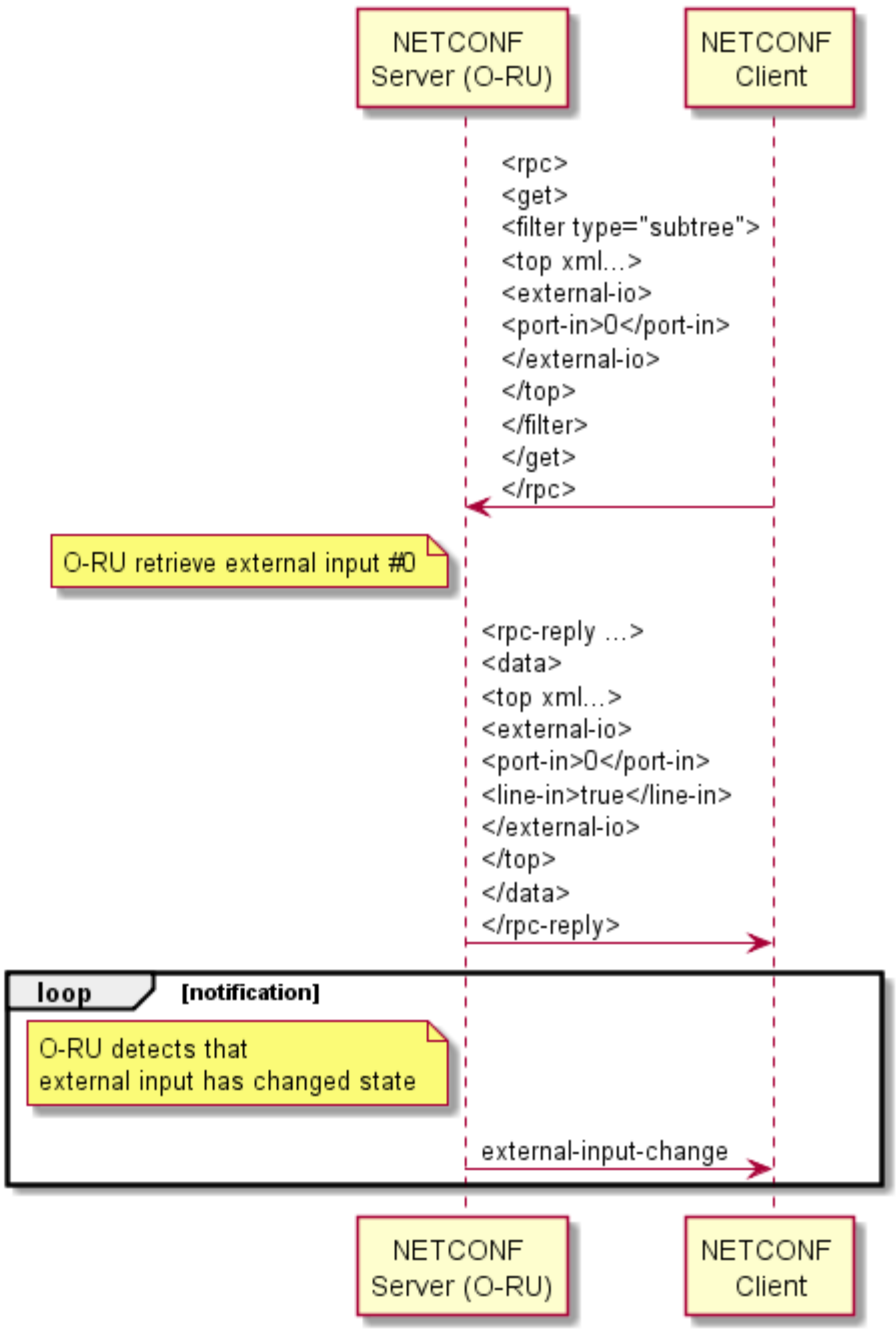


Figure 41**– retrieve external line-in**

11.4.2 External output

This section explains single external output line case. For multiple external outputs case, same behaviour for each output shall be processed individually.

For output, the O-RU and O-RU controller shall support two scenarios.

[1] To retrieve output state from O-RU controller and respond the output state from the O-RU to O-RU controller.

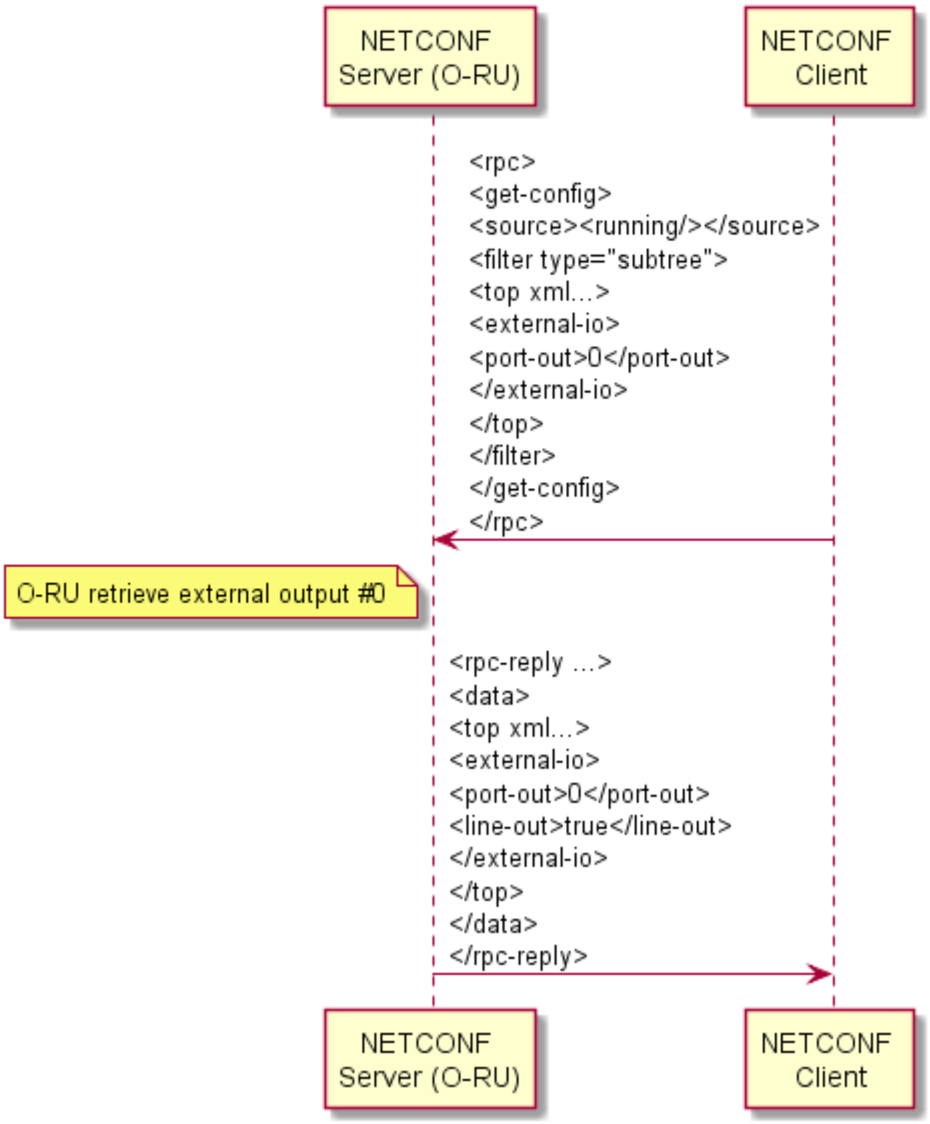
[2] To send edit-config from the O-RU to O-RU controller when output state change is required.

The value shall be

TRUE: Circuit is open.

FALSE: Circuit is closed

The default values shall be TRUE.



```
<rpc>
<get-config>
<source><running/></source>
<filter type="subtree">
<top xml...>
<external-io>
<port-out>0</port-out>
</external-io>
</top>
</filter>
</get-config>
</rpc>
```

```
<rpc-reply ...>
<data>
<top xml...>
<external-io>
<port-out>0</port-out>
<line-out>true</line-out>
</external-io>
</top>
</data>
</rpc-reply>
```

Figure 42: retrieve external line-out

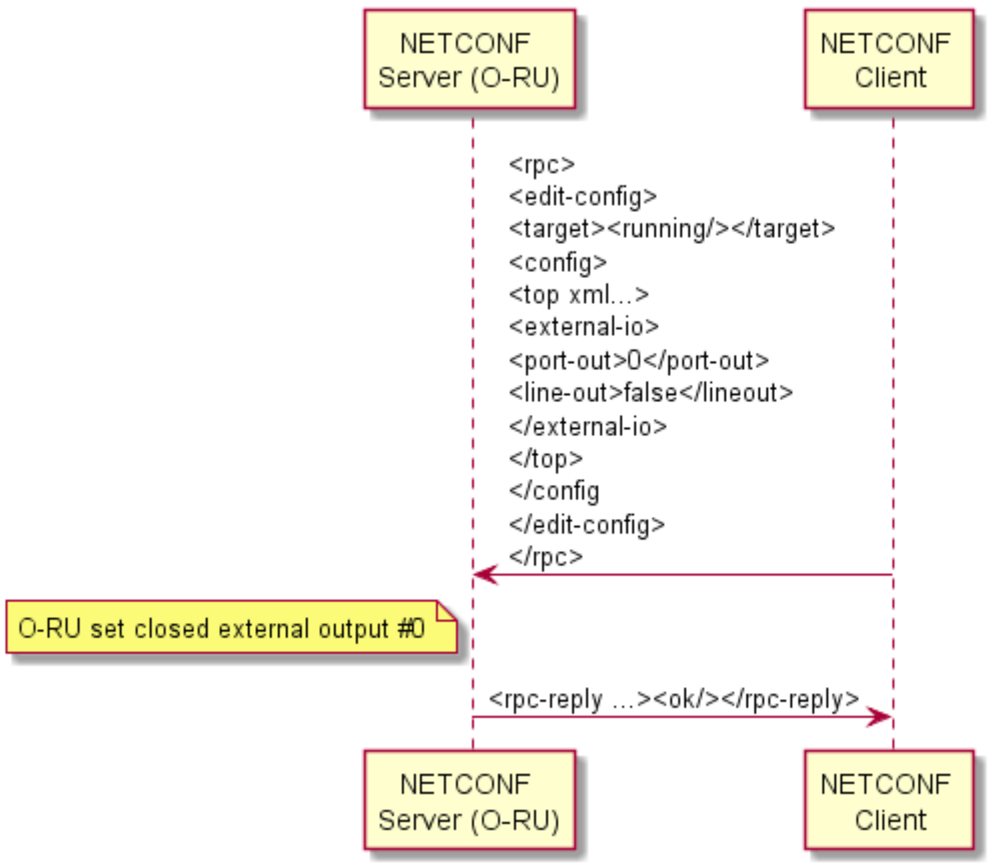


Figure 43: **control external line-out**

Chapter 12 Details of O-RU Operations

12.1 Retrieval of O-RU Information

This sub-section provides handling for O-RU controller(s) to retrieve O-RU information from O-RU. The further actions such as SW file management, U-plane configuration and Performance Management will use these retrieved O-RU information.

The following information, for example, can be retrieved from the O-RU:

hw/hardware/component

retrieve **mfg-name** – the name of the O-RU manufacturer

retrieve **serial-num** – the serial number of the O-RU

retrieve **software-rev** – the version of the O-RU software build

o-ran-hardware/hardware/component/

retrieve **product-code** – the O-RAN defined product code

o-ran-operations/operational-info/declarations

retrieve supported-mplane-version – the version of the O-RAN M-Plane interface

retrieve **supported-cusplane-version** – the version of the O-RAN CUS-Plane interface

retrieve **supported-header-mechanism** – the type of C/U plane headers supported by the O-RU o-ran-operations/operational-state

retrieve **restart-cause** – the reason for the last restart

o-ran-sync/sync

retrieve **sync-state** – the synchronization state of the O-RU

The detail of O-RU information, please see corresponding YANG modules in Annex D.

12.2 User plane message routing

The purpose of U-Plane configuration is to define the relationship between U-Plane application endpoints in the O-DU and those in the O-RU. After such relationships are defined, the application endpoints are able to exchange IQ data using the U-Plane application protocol defined in [2].

Precondition:

- M-Plane connectivity is established between NETCONF Client and NETCONF Server

12.2.1 Configurable format for eAxC_ID

The eAxC_ID is used by C/U-plane application to manage eCPRI communication between desired C/U-Plane application components in O-DU and O-RU.

As defined in [2], the eAxC_ID consists of four parameters: DU-PORT, RU-PORT, CC-ID and BAND-SECTOR-ID. Order of parameters in eAxC_ID must follow definitions in CUS-Plane spec. In this version of the O-RAN WG4 specification, the length of eAxC_ID is constant and equal to 16 bits. To enable optimal sharing of the 16 bits between these four parameters, the assignment of eAxC_ID bits to parameters is not fixed. As a consequence, there is a need for NETCONF client to configure the bit assignment to parameters mappings using the M-Plane interface.

To handle flexible bit assignment, configurable bitmasks are defined for each parameter.

Note: flexible configuration means, that bits of eAxC_ID can be assigned to parameters in runtime.

Rules to be followed by NETCONF Client when configuring bit assignments:

- notation used for parameters forming eAxC_ID is from the LSB.

- each parameter uses consecutive bits

- each parameter can occupy 0-16 bits

- single bit of eAxC_ID cannot be assigned to more than one parameter

RPC **edit-config** shall be used to configure bit assignments to O-RU.

Bit assignment change for parameters related to an existing carrier is not allowed. (Impacted carriers need to be deactivated and deleted prior to any change in eAxC_ID configuration, and then be subsequently created and activated.)

An example of bit assignment usage where 3 bits are assigned to the BAND-SECTOR-ID, 3 bits for CC-ID, 1 bit for DU-PORT-ID and 3 bits for RU-PORT-ID is shown below:

```
        <du-port-bitmask> 1111111000000000 </du-port-bitmask>
    <band-sector-bitmask> 0000000111000000 </band-sector-bitmask>
           <ccid-bitmask> 0000000000111000 </ccid-bitmask>
        <ru-port-bitmask> 0000000000000111 </ru-port-bitmask>
```

12.2.2 U-Plane endpoint addressing

Parameter “eaxc-id” for low-level-tx-endpoint and low-level-rx-endpoint, defined using an unsigned 16 bit integer, shall follow the eAxC_ID addressing schema defined in section 12.2.1.

The NETCONF Client is responsible for assigning unique values to "eaxc_id" addresses of all low-level-rx-endpoint elements and low-level-tx-endpoint elements, that are linked to the same interface element and are referenced by low-level-rx-link or/ low-level-tx-link respectively. Note: The same eAxC_id can be assigned to low-level-rx-endpoint and low-level-tx-endpoint.

The NETCONF Server shall reject configuration requests violating the above-mentioned rules.

The default value of the "eaxc-id" parameters is zero.

12.2.3 General configuration scenario

Below is described the general scenario to be followed by a NETCONF Client in order to properly configure communication between C/U-Plane endpoints in the O-DU and O-RU.

All operations can be performed in any order (including combining some of them in one request) provided assumed result (overall configuration) of each request sent by NETCONF Client is valid. Note selected highlighted rules below:

- eaxc-id is unique for all endpoints linked with same interface element and linked with any low-level-rx-link or low-level-tx-link element

- at the moment of creation, every low-level-rx-link must be linked to an existing rx-array-carrier element and existing processing-element element

- at the moment of creation, every low-level-tx-link must be linked to an existing tx-array-carrier element and existing processing-element element

1) NETCONF Client determines the presence of following elements offered by NETCONF Server:

- tx-arrays - by fetching the list of **tx-arrays** in o-ran-uplane-conf.yang
- rx-arrays - by fetching the list of **rx-arrays** in o-ran-uplane-conf.yang
- low-level-tx-endpoint elements - by fetching the list **static-low-level-tx-endpoints** in o-ran-uplane-conf.yang
- low-level-rx-endpoint elements - by fetching the list **static-low-level-rx-endpoints** in o-ran-uplane-conf.yang
- interface elements - by fetching list of **interfaces** in o-ran-interfaces.yang

2) NETCONF Client determines capabilities exposed by static-low-level-tx-and points and static-low-level-rx-endpoints. Additionally. NETCONF Client determines capabilities exposed by "endpoint-types" and "endpoint-capacity-sharing-groups" and specific parameters proprietary to [tr]x-array(s). Obtained information must be respected when NETCONF Client configures low-level-[tr]x-endpoints refferenced to static-low-level[tr]x-endpoints by parameter "name".

3) For elements determined in step 1) NETCONF Client examines relationship between

- static-low-level-tx-endpoint elements and tx-array elements in o-ran-uplane-conf.yang
- static-low-level-rx-endpoint elements and rx-array elements in o-ran-uplane-conf.yang
- static-low-level-tx-endpoint elements and interface elements
- static-low-level-rx-endpoint elements and interface elements
- tx-arrays, rx-arrays and their elements in o-ran-uplane-conf.yang.

Note: Netconf Client retrievies the content of o-ran-beamforming,yang module to obtain knowledge regarding beamforming-related parameters that apply for particular Netconf Server. This step is optional, as o-ran-beamforming.yang module exists only in case Netconf Server supports beamforming. Obtained parameters are needed by Netconf Client to perform beamforming control.

4) For every static-low-level-rx-endpoint NETCONF Client determines endpoint's ability to support non-time managed and/or time managed traffic. Information about delayed traffic type supported by endpoints is exposed through parameter **delayed-data-transfer-supported** (Boolean) under static-low-level-rx-endpoint. In case endpoint supports non-time managed traffic - it is assumed that also time-managed traffic is supported by the endpoint (time-managed traffic support is mandatory function whilst non-time managed traffic support is optional function), the endpoint requires desired type of supported traffic to be configured to the endpoint. Configuration is assumed to be static for run-time. Configuration is applicable with "**non-time-managed-delay-enabled**" (Boolean) parameter of low-level-rx-endpoint related by "name" to static-low-level-rx-endpoint exposing endpoints ability to support non-time managed traffic. Default value of this parameter is FALSE, meaning endpoint supports time managed traffic by default. For details see: Note 2.

5) NETCONF Client may optionally perform “Transport connectivity checking procedure” to NETCONF Server, as described in section 4.6.

Note: After this step is successfully completed, a NETCONF Client knows which low-level-tx-endpoint elements and low-level-rx-endpoint elements offered by the NETCONF Server are accessible for TX/RX endpoints served by the NETCONF Client.

6) NETCONF Client determines accessible low-level-rx-endpoint elements and low-level-tx-endpoint elements, that are suitable for the desired cell configuration (i.e. are linked with specific antenna arrays and are able to support desired type of traffic).

7 NETCONF Client assigns unique values to the eaxc-id(s) to the endpoints determined in step 1.

Note: uniqueness of eaxc-id is mandatory within all endpoints related to same interface elements and having relationship to low-level-rx-endpoint elements or low-level-tx-endpoint elements.

Note 2: In case NETCONF Client wants particular value of eAxC_IDto be used for non-time managed traffic, NETCONF Client shall assign this eAxC_ID to parameter "eaxc-id" belonging to low-lever-rx-endpoint, that is capable to support non-time managed traffic (as per reference to capabilities exposed by corresponding static-low-level-rx-endpoint corresponding to low-level-rx-endpoint by name). When assigning eAxC_ID to convinient low-level-rx-endpoint, NETCONF Client must also configure the low-level-rx-endpoint to work in non-time-managed mode (when applicable - see: 4)). Change between types of traffic configured to low-level-rx-endpoint shall not be requested by NETCONF Client in case there is traffic served by endpoint that is a subject of reconfiguration.

8) NETCONF Client creates **tx-array-carrier**(s) and **rx-array-carrier**(s)

9) NETCONF Client creates **processing-elements** related to **interfaces** offering access to desired endpoints (suitable in terms of capabilities and able to process signals related with desired [tr]x-array)

10) NETCONF Client creates low-level-[tr]x-link(s) to make relationship between low-level-[tr]x-endpoint(s), [tr]x-array-carriers and processing elements belonging to transport. Respective TX path and RX path linkage must be followed.

Note: C/U-Plane traffic can be prioritized by reference "uplane-flow-name" indicated by low-level-rx-link in o-ran-uplane-conf.yang. The reference is to o-ran-processing-element.yang, where it is linked to "uplane-marking". Further the uplane-marking points to o-ran-interfaces.yang, where it ends up pointing to priority depending on actually used u-plane transport (either PCP for Ethernet ot DCSP for IP). For details regarding priorities see: ORAN-WG4.CUS.0-v01.00, section 3.3 "Quality of Service".

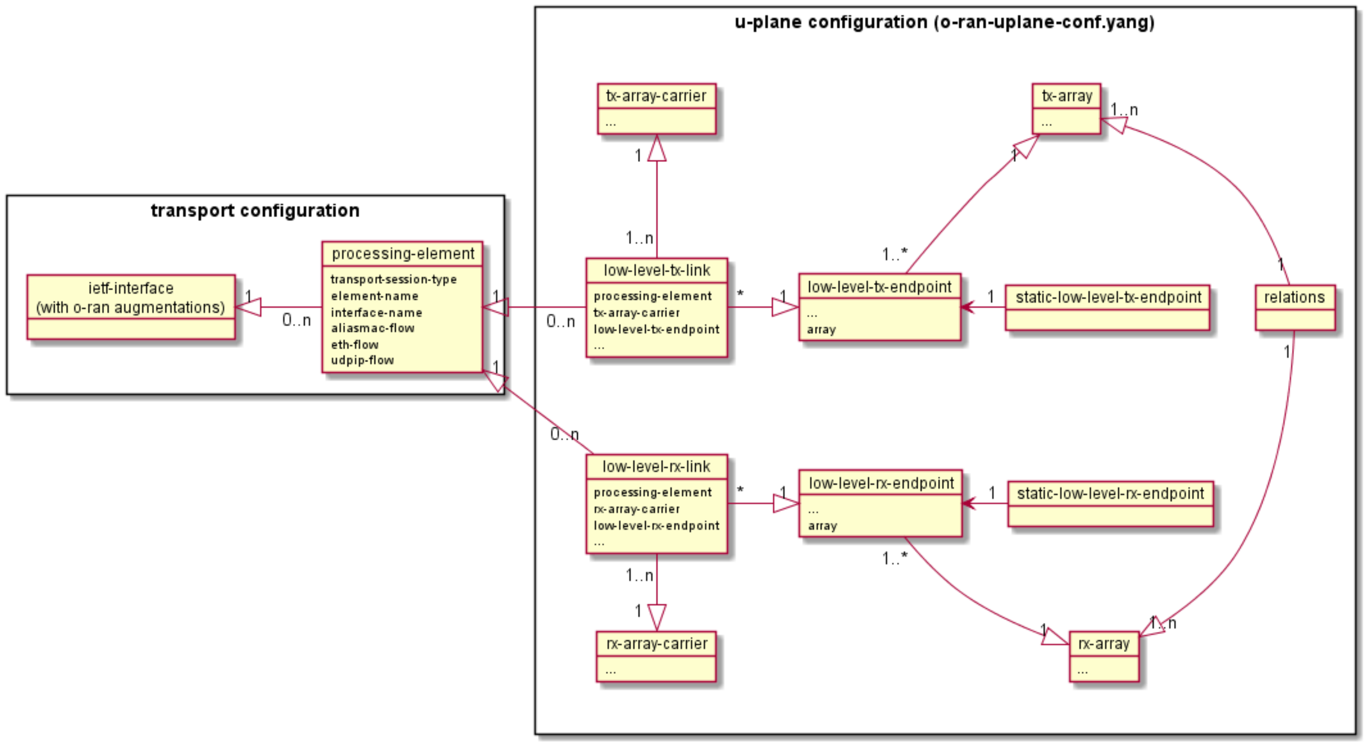


Figure 44 Diagram showing relations between CU-Plane and Carrier configuration elements

For detailed content of objects shown in "u-plane configuration" box on above diagram, please examine o-ran-uplane-conf.yang module.

After steps above carrier configuration scenario can be started. This is described in subsection 12.3.

12.3 Carrier Configuration

12.3.1 Carrier creation

This section provides basic scenario for carrier creation procedure. Precondition for below steps is to fulfil steps from subsection 12.2.3

1) NETCONF Client creates the **tx-array-carriers** in relation to the desired **tx-arrays**. Note that generally number of **tx-array-carriers** will be the same as multiple of the desired number of **tx-arrays** and the number of component carriers.

2) NETCONF Client creates the **rx-array-carriers** in relation to the desired **rx-arrays**. Note that generally number of **rx-array-carriers** will be the same as multiple of the desired number of **rx-arrays** and the number of component carriers.

3) NETCONF Client creates the **processing-elements** related to **interfaces** offering access to endpoints.

4) NETCONF Client creates the **low-level-tx-links** containing relationship to the existing **tx-array-carriers**, **low-level-tx-endpoints** and existing **processing-elements**.

5) NETCONF Client creates the **low-level-rx-links** containing the relationship to existing **rx-array-carriers**, **low-level-rx-endpoints** and existing **processing-elements**.

With the above steps successfully performed, the relationship between C/U-Plane application endpoints at O-DU and O-RU is configured.

12.3.2 Activation, deactivation and sleep

The NETCONF Client performs activation by setting the value of the parameter “active” at tx-array-carrier element / rx-array-carrier element to “ACTIVE”.

The NETCONF Client performs deactivation by setting the value of the parameter “active” at tx-array-carrier element / rx-array-carrier element to “INACTIVE”

Communication between related U-Plane endpoints is enabled under condition, that for corresponding tx-array-carrier or rx-array-carrier value of parameter “active” is “ACTIVE” and value of parameter “state” is “READY”. Otherwise communication is disabled.

The NETCONF Client can put the tx-array-carrier / rx-array-carrier to sleep by setting value of parameter “active” in the corresponding tx-array-carrier element / rx-array-carrier element to “SLEEP”.

A particular tx-array-carrier / rx-array-carrier is in sleep mode when value of its parameter “active” is “SLEEP” and value of its parameter “state” is “READY”.

Below diagram shows possible transitions to be followed by "active" and "state".

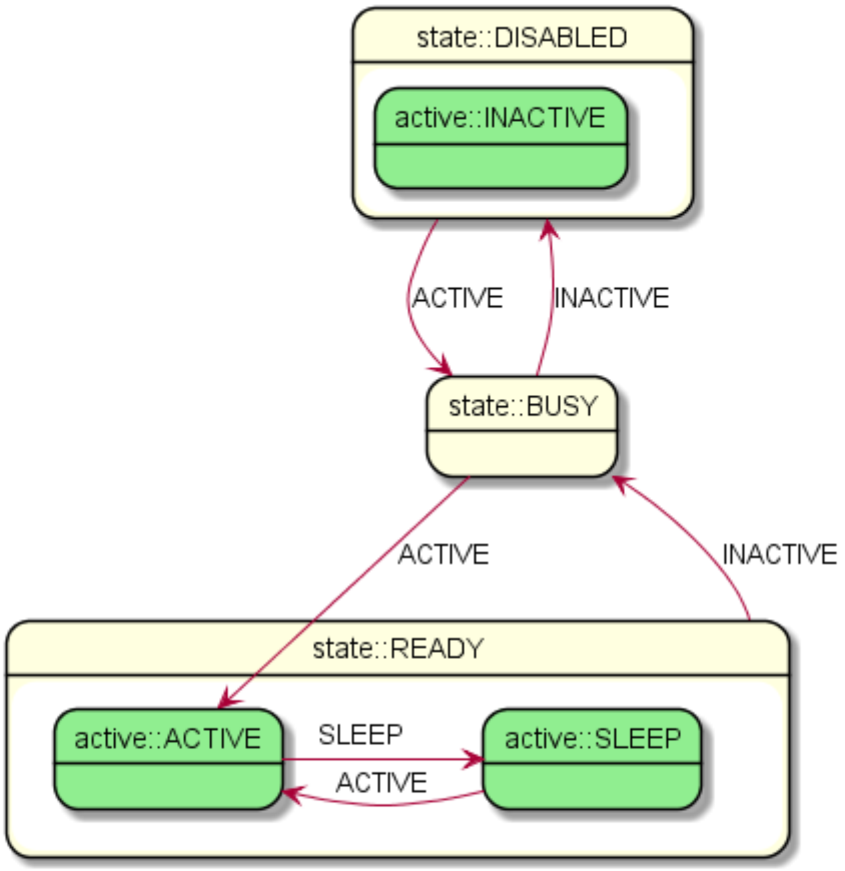


Figure 45: Diagram showing possible transitions of "active" and "state"

12.4 Beamforming Configuration

The beamforming functionality allows the O-RU to influence the angle of the main lobe of the signal which is radiated from/received by the O-RU. Beamforming support is optional and an O-RU shall indicate that it supports such functionality by indicating that it supports the “urn:o-ran:beamforming:x.y” namespace.

A multi-band capable O-RU shall be able to support independent beamforming configuration on each of its supported bands.

12.4.1 Default Beamforming Configuration

In case the O-RU supports beamforming, the o-ran-beamforming.yang and o-ran-uplane-conf,yang modules are used to report the default relationship between supported beams to a NETCONF client. A **band-number** is used to uniquely identify separate bands supported by a multi-band O-RU with the beamforming configuration referencing the set of **tx-arrays** and **rx-arrays** that are associated with this band.

12.4.2 Beamforming Configuration Update

This section provides the method to modify and to apply the beamforming configuration (weights, attributes and/or beam properties). The modification of the beamforming information is allowed only if O-RU supports the feature “MODIFY-BF-CONFIG” used for defining the modification of beamforming configuration.

The beamforming configuration is stored in the O-RU and comes from the O-RU’s software, treated in Chapter 5. The O-RU shall locate the beamforming configuration file in the generic folder, i.e., O-RAN/beamforming/.

To modify the beamforming configuration, the following steps are applied.

1. NETCONF client can retrieve the file list of the O-RU’s folder: O-RAN/beamforming/.
2. NETCONF client can trigger the upload of the beamforming configuration file from the O-RU’s folder.
3. Operator can recover the uploaded file and edit the beamforming configuration file offline.
4. NETCONF client can download the file to the original folder.

The modified beamforming configuration file shall not have the same name as any other file in the folder. Its file name is the matter of implementation.

The beam properties in -ran-beamforming YANG module contain **coarse-fine**, **coarse-fine-beam-relation** and **neighbor-beam** for each **beam-id**. This information is received from the O-RU as O-RU's capability at O-RU start-up and typically are used by the scheduler in O-DU. A NETCONF client (O-DU or NMS) can modify the beamforming information via file described in this section. When the beamforming configuration (weight, attribute and beam properties) is modified via file, the configuration of the beam properties list in the o-ran-beamforming YANG module should be modified together via the same file if affected by the modified weight and/or attribute.

An O-RU supporting the modification of beamforming configuration shall support the storage of at least two beamforming files per simultaneous band supported. For each band within a multi-band O-RU, one file corresponds to the default (factory, read-only) beamforming configuration and at least one file corresponds to a modified (read-write) beamforming file. The O-RU has the responsibility to remove existing file and prepares space for new file when the NETCONF client **file-download** rpc is issued. When the O-RU only supports the storage of a single modified (read-write) beamforming file per band of operation, i.e., **number-of-writable-beamforming-files** = 1 the **file-download** operation for the modified beamforming configuration needs to be done while neither **tx-array-carriers** nor **rx-array-carriers** are configured in the O-RU to avoid the removal of the modified beamforming configuration file for the current active software.

If the O-RU supports the capability to store two or more modified beamforming configuration files per band of operation in the O-RU, i.e., **number-of-writable-beamforming-files** > 1, the NETCONF **file-download** operation can be performed without any timing limitation. That's because the modified beamforming configuration file for the current beamforming configuration can be kept during the **file-download** operation. To apply the new modified beamforming configuration, the following steps are applied:

1. The NETCONF client can download the file to the beamforming folder if the O-RU supports the capability **number-of-writable-beamforming-files** > 1.
2. The NETCONF client shall deactivate **tx-array-carriers** and **rx-array-carriers** in the U-Plane configuration by setting "INACTIVE" for the **active** parameters, if they are ACTIVE.
3. Optionally, the NETCONF client shall delete **tx-array-carriers** and **rx-array-carriers**, if O-RU doesn't support the capability **update-bf-non-delete**.
4. Alternatively, the NETCONF client can trigger the download of the modified beamforming configuration file to the folder if the O-RU's capability is **number-of-writable-beamforming-files** =1.
5. The NETCONF client shall activate the modified beamforming configuration by using **activate-beamforming-config** rpc and selecting the modified beamforming configuration file and the **band-number** for which this modified configuration applies.
6. If a NETCONF client subscribes to the notification beamforming-update in advance, the O-RU sends a notification beamforming-update to the subscribed NETCONF client. Then the NETCONF client can subsequently retrieve beam properties in o-ran-beamforming YANG module via NETCONF <**get**> operation.
7. Optionally, the NETCONF client shall create **tx-array-carriers** and **rx-array-carriers** again, if the O-RU doesn't support the capability **update-bf-non-delete**.

8. The NETCONF Client shall activate **tx-array-carriers** and **rx-array-carriers** in the U-Plane configuration by setting "ACTIVE" for **active** parameters.

Then the new edited beamforming information is applied to the new **tx-array-carriers** and **rx-array-carriers** in the U-Plane configuration.

[Abnormal handling] If the O-RU fails to activate the edited beamforming configuration file correctly, i.e., rpc error for rpc **activate-beamforming-config**, the O-RU shall revert back to the default beamforming configuration file and report this to the NETCONF client.

At the **reset** rpc, the beamforming configuration information is switched to the default beamforming configuration. Even though the reset operation is issued, the O-RU may store the modified beamforming configuration file in the folder, which is not used, if O-RU supports the capability **persistent-bf-files** to store them in the reset-persistent memory.

The file format of the beamforming configuration is O-RU implementation specific.

The diagrams below show two methods to modify the file of beamforming configuration information plus the method how to apply the modified file for beamforming configuration conformation to use.

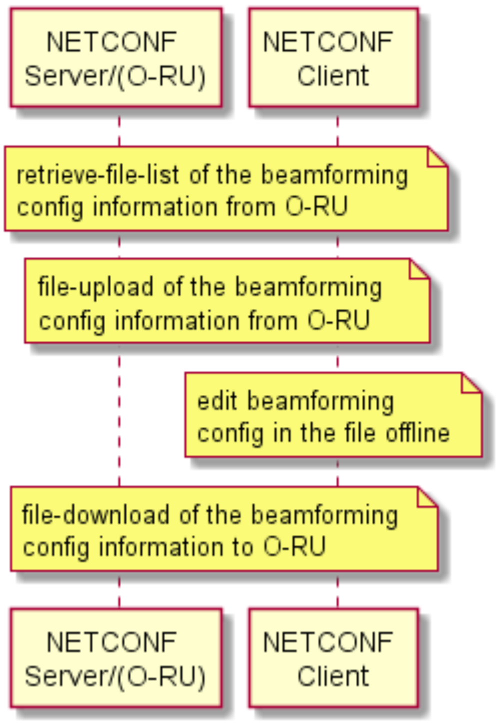
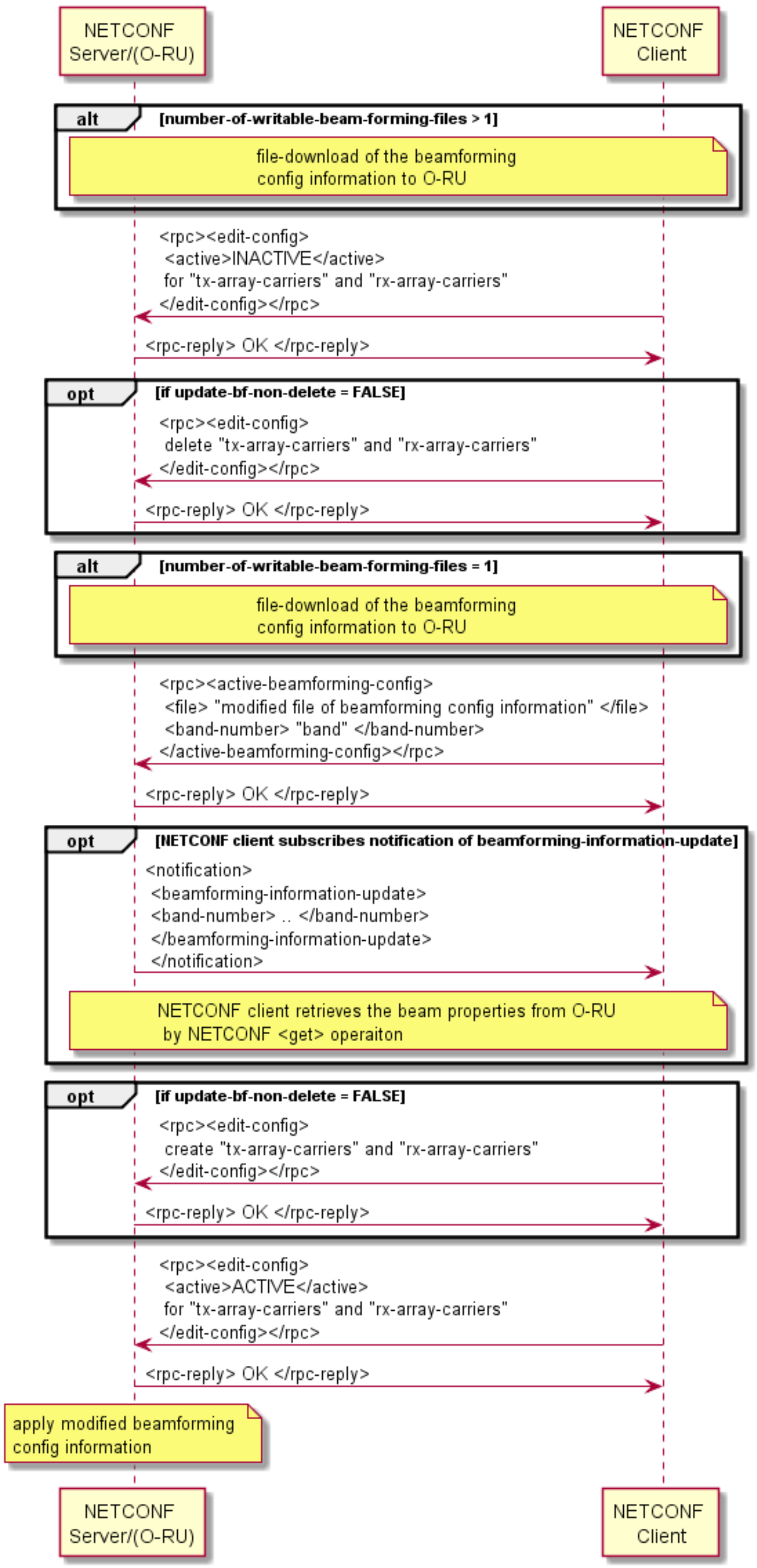

NETCONF Server/(O-RU)
NETCONF Client
retrieve-file-list of the beamforming config information from O-RU
file-upload of the beamforming config information from O-RU
edit beamforming config in the file offline
file-download of the beamforming config information to O-RU
NETCONF Server/(O-RU)
NETCONF Client

Figure 46: Method to Modify the File of Beamforming Configuration Information


NETCONF Server/(O-RU)
NETCONF Client
alt [number-of-writable-beam-forming-files > 1]
file-download of the beamforming config information to O-RU
<rpc><edit-config> <active>INACTIVE</active> for "tx-array-carriers" and "rx-array-carriers" </edit-config></rpc>
<rpc-reply> OK </rpc-reply>
opt [if update-bf-non-delete = FALSE]
<rpc><edit-config> delete "tx-array-carriers" and "rx-array-carriers" </edit-config></rpc>
<rpc-reply> OK </rpc-reply>
alt [number-of-writable-beam-forming-files = 1]
file-download of the beamforming config information to O-RU
<rpc><active-beamforming-config> <file> "modified file of beamforming config information" </file> <band-number> "band" </band-number> </active-beamforming-config></rpc>
<rpc-reply> OK </rpc-reply>
opt [NETCONF client subscribes notification of beamforming-information-update]
<notification> <beamforming-information-update> <band-number> .. </band-number> </beamforming-information-update> </notification>
NETCONF client retrieves the beam properties from O-RU by NETCONF <get> operaiton
opt [if update-bf-non-delete = FALSE]
<rpc><edit-config> create "tx-array-carriers" and "rx-array-carriers" </edit-config></rpc>
<rpc-reply> OK </rpc-reply>
<rpc><edit-config> <active>ACTIVE</active> for "tx-array-carriers" and "rx-array-carriers" </edit-config></rpc>
<rpc-reply> OK </rpc-reply>
apply modified beamforming config information
NETCONF Server/(O-RU)
NETCONF Client

1

2

Figure 47: Method to Apply the modified file for Beamforming Configuration Information

12.4.3 Dynamic Beamforming Control option

As option, O-RU may support dynamic beamforming control mode. Support for this type of beamforming control can be recognized—in the case of weights-based dynamic beamforming—from value of parameter rt-bf-weights-update-support (TRUE), and—in the case of attributes-based dynamic beamforming—from the value of parameter rt-bf-attributes-update-support (TRUE), in o-ran-beamforming.yang module

In dynamic beamforming control mode DU updates content of lookup table in O-RU using eCPRI C-Plane messages. For details of eCPRI messaging please see ORAN-WG4.CUS.0-v01.00 spec, chapter 5.4.2 "Scheduling and Beamforming Commands".

Dynamically updated content of lookup table is further addressed by DU in the same way as it is done for static beamforming - by requesting particular Beam ID to be applied.

In case dynamic beamforming control is supported, O-RU indicates following supplementary information using parent leaf "static-properties" in o-ran-beamforming.yang module.

- beamforming type (frequency domain, time domain, hybrid)

- beamforming weight compression format (optional)

- available range of Beam IDs, that can be dynamically updated by DU.

- supported time and frequency granularity for time domain and hybrid beamforming control.

Note: Neighborhood relations between beams produced by beam IDs controlled by DU are unknown to O-RU, hence are not exposed.

In the case of weights-based dynamic beamforming, to properly calculate beamforming weights DU needs to know antena array geometry. This information DU obtains by reading the content of o-ran-uplane-conf.yang (list of tx-arrays and rx-arrays with their child parameters). Details of beamforming weight calculations are not a subject for M-Plane activity and as such are intentionally not covered in this specification.

Chapter 13 Licensed-Assisted Access

13.1 Introduction:

Licensed-assisted access (LAA) leverages the carrier-aggregation (CA) functionality. With LAA, CA is performed between licensed and unlicensed component carriers (CCs). This enables the LAA system to opportunistically benefit from using unlicensed spectrum (e.g., UNII bands in the 5 GHz spectrum) to enhance the aggregated capacity of the O-RU with the objective of enhancing the downlink throughput.

Several modifications in the RAN are needed to enable LAA in the O-DU and O-RU such as listen-before-talk (LBT), discontinuous transmission, carrier-selection, discovery reference signal (DRS) transmission, etc. This section is focused on the LAA-related messages and procedures needed in the M-plane. The C-plane related messages are defined in the CUS-plane spec [2].

This version of the M-plane spec supports only LAA based on Rel. 13 of the 3GPP specs, where transmission on the unlicensed spectrum can be done only in the downlink direction. The support of eLAA Rel. 14 (i.e., enabling UL transmission on the unlicensed spectrum) may be included in a later version of the M-plane spec.

The modifications at M-plane to Support LAA can be summarized as follows:

i) LAA-initiation process: O-DU learns about O-RU capabilities and configures it.

- O-RU LAA Support: The O-RU indicates it supports LAA by including support for the "urn:o-ran:laa:x.y" and "urn:o-ran:laa-operations:x.y" namespaces in its ietf-yang-library model (RFC 7895) [20].
- O-RU LAA Capability Information: When the LAA feature is enabled, leafs corresponding to LAA-related O-RU capabilities, such as the number of supported LAA SCarriers, maximum LAA buffer size, etc, are conveyed via the M-plane to the O-DU as part of the o-ran-module-cap.yang module.
- O-RU LAA Configuration: The NETCONF client configures the unlicensed LAA component carrier with the LAA-related parameters such as the energy-detection threshold, DRS measurement timing configuration (DMTC) period, etc. as part of the o-ran-uplane-config.yang module. The configuration of the number of LAA SCarriers, multi-carrier type, etc. is performed using the o-ran-laa.yang Module.
- For explanation of the LAA-initiation process, please refer to Figure 5 in Chapter 3, where the O-RU LAA capability info is conveyed within the “Retrieval of O-RU information” step, while the O-RU LAA configurations are conveyed in “Configuring the O-RU operational parameters” step in Figure 5.

ii) Carrier-selection: Selecting the best channel in the unlicensed band, both initially and dynamically over time.

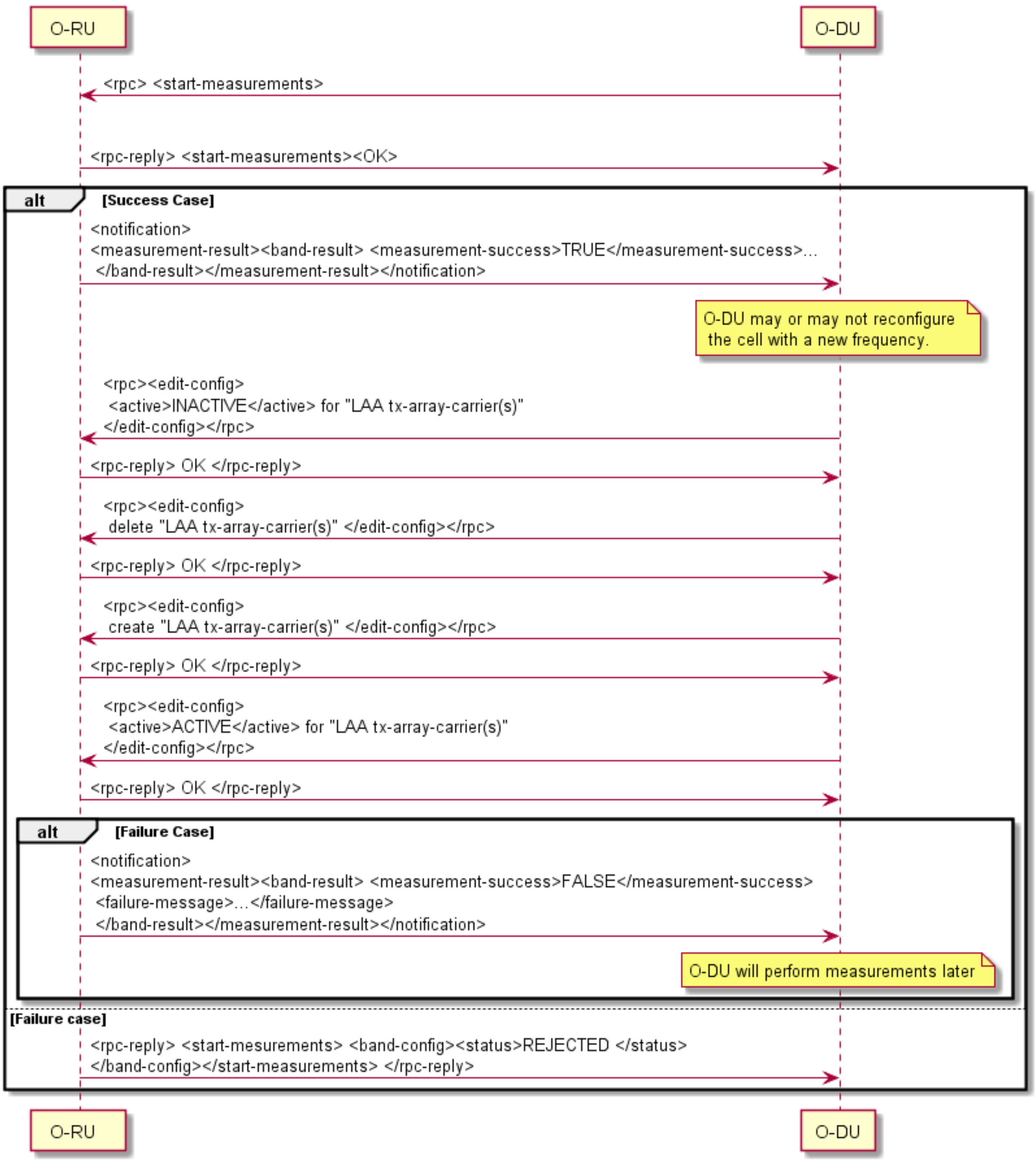

O-RU
O-DU
<rpc> <start-measurements>
<rpc-reply> <start-measurements><OK>
alt
[Success Case]
<notification> <measurement-result><band-result> <measurement-success>TRUE</measurement-success>... </band-result></measurement-result></notification>
O-DU may or may not reconfigure the cell with a new frequency.
<rpc><edit-config> <active>INACTIVE</active> for "LAA tx-array-carrier(s)" </edit-config></rpc>
<rpc-reply> OK </rpc-reply>
<rpc><edit-config> delete "LAA tx-array-carrier(s)" </edit-config></rpc>
<rpc-reply> OK </rpc-reply>
<rpc><edit-config> create "LAA tx-array-carrier(s)" </edit-config></rpc>
<rpc-reply> OK </rpc-reply>
<rpc><edit-config> <active>ACTIVE</active> for "LAA tx-array-carrier(s)" </edit-config></rpc>
<rpc-reply> OK </rpc-reply>
alt
[Failure Case]
<notification> <measurement-result><band-result> <measurement-success>FALSE</measurement-success> <failure-message>...</failure-message> </band-result></measurement-result></notification>
O-DU will perform measurements later
[Failure case]
<rpc-reply> <start-mesurements> <band-config><status>REJECTED </status> </band-config></start-measurements> </rpc-reply>
O-RU
O-DU

Figure 48: Carrier-Selection Call-flow

13.2 LAA-initiation Process

13.2.1 LAA Module Capabilities

During LAA-initiation, the O-RU reports its LAA capabilities to the NETCONF client. These capabilities are sent at the start up as part of the o-ran-module-cap.yang module. The attributes included are:

(1) **sub-band-frequency-ranges**: The unlicensed sub-bands (e.g., 46A, 46B, etc.) that are supported at the O-RU and their frequency ranges

(2) **number-of-laa-scarriers** (uint8): Number of LAA Scarriers that the O-RU can support.

(3) **maximum-laa-buffer-size** (uint16): Maximum O-RU buffer size in Kilobytes (KB) per CC. This parameter is needed at the O-DU to know how much data can be sent in advance and stored at the O-RU to address the LBT uncertainty.

(4) **maximum-processing-time** (uint16): Maximum O-RU Processing time in microseconds at the O-RU to handle the received/transmitted packets from/to the O-DU. This parameter is needed at the O-DU to determine the time where it needs to send the data to the O-RU.

(5) **self-configure** (Boolean): Capability to manage the contention window at the O-RU. Based on the CUS-spec, there are two modes of operation for LAA, 1) when the contention window is managed by the O-DU, and 2) when the contention window is managed by the O-RU. This field is set to True if the O-RU can manage the contention window locally.

13.2.2 LAA O-RU Parameter Configuration

The second stage of the LAA-initiation process is the configuration message (using RPC edit-config). In this message, the O-DU configures the O-RU with the required parameters in the downlink direction. LAA parameters can be configured by Netconf Client after capability exchange is finished. It can also be sent as needed, to reconfigure the O-RU with new parameters (e.g., **ed-threshold-pdsch**, etc.). The attributes of this message (o-ran-laa.yang Module) include:

(1) **number-of-laa-scarriers** (uint8): Number of LAA SCarriers to be used at the O-RU. This number should be less than or equal the number reported by the O-RU in its module capabilities.

(2) **multi-carrier-type** (Enumeration): This value indicates the list of multi carrier types (A1, A2, B1, B2) which as subsection 15.1.5 in [32].

(3) **multi-carrier-tx** (Boolean): This value indicates whether self-deferral is activated or not. “True” indicates transmission on channel access win (i.e., no self-deferral). “False” indicates mutual transmission on multiple carriers.

(4) **multi-carrier-freeze** (Boolean): This value indicates if the absence of other technology in the unlicensed band can be guaranteed. This attribute can only be used when the multi-carrier-type is A1. “False” indicates that absence of other technology is not guaranteed.

(5) **laa-ending-dwpts-supported** (Boolean): This value indicates whether LAA ending in Downlink Pilot Time Slot (DwPTS) is supported.

(6) **laa-starting-in-second-slot-supported** (Boolean): This value indicates LAA starting in second slot is supported.

LAA carrier configurations (o-ran-uplane-conf.yang Module) include:

(1) **ed-threshold-pdsch** (int8): This value indicates the energy detection (ED) threshold for LBT for PDSCH and for measurements in dBm.

(2) **ed-threshold-drs** (int8): This value indicates the ED threshold for LBT for DRS in dBm.

(4) **tx-antenna-ports** (uint8): This value indicates the Tx antenna ports for DRS.

(5) **transmission-power-for-drs** (int8): This value indicates the offset of CRS power to reference signal power (dB).

(6) **dmtc-period** (enumeration): This value indicates DMTC period in milliseconds.

(7) **dmtc-offset** (uint8): This value indicates DMTC offset in Subframes.

(8) **lbt-timer** (uint16): This value indicates LBT Timer in milliseconds.

If Self Configure capability is set to “true”, the following parameters are also needed to be configured. For every traffic priority class, the O-DU needs to configure maximum CW usage counter. This value indicates the maximum value of counter which shows how many max congestion window value is used for back off number of each priority class traffic. This value is defined at section 15.1.3 of [32] as K. Based on the 3GPP specification, this value is selected by O-RU from the set of values {1, 2, …, 8}

13.3 Carrier-Selection

13.3.1 LAA Measurements

The function of the message “**rpc start-measurements**” is to order the O-RU to start measurements. This message can be used for carrier selection initially or dynamically over time. O-RU sends RPC response where status==ACCEPTED (positive case) or REJECTED (negative case). O-RU performs measurement and delivers result with respect to max response time. If result is not ready on time - O-RU sends notification with "measurement-success" == FALSE and with appropriate failure reason. For every configured band, the O-RU informs the NETCONF client whether the measurement was successful or not. For bands with successful measurements, the O-RU reports the occupancy ratio and average RSSI for each channel. For bands with failure measurements, the O-RU includes the reason (e.g., TIMEOUT when the O-RU is not able to finish the measurement for this specific band).

The occupancy ratio of a given channel is defined as the percentage of the busy duration (i.e., measured signal power is larger than the energy-detection threshold) to the total measurement duration of this specific channel. The energy-detection threshold is specified in the o-ran-uplan-conf.yang module using the **ed-threshold-pdsch** leaf. Note that this threshold is the same as the energy-detection threshold used for LBT for PDSCH transmission. The total measurement duration per channel is specified in o-ran-laa-operation module using the **duration-per-channel** leaf. The range of the occupancy ratio is from 0% (no signal is detected over the total measurement duration per channel) to 100% (i.e., channel was always occupied during measurement).

The average RSSI of the measured channel is the measured power of this specific channel averaged over the total measurement duration per channel. This parameter is reported to the NETCONF client in dBm and takes a value from the range 0 dBm to -128 dBm.

13.3.2 LAA Carrier Frequency Configuration

After receiving the measurements, the NETCONF client configures the O-RU with the new channel(s), if needed. For every component carrier (CC) that needs to be configured with a new center frequency, the O-DU will need to first deactivate the TX carrier, delete it, then create a new TX carrier (with the new center carrier frequency as well as any other new configurations), and then activate the TX carrier again to start OTA operation. The procedure for deactivating/deleting/creating/activating the carrier is explained in Section 12.3: Carrier Configuration, in the M-plane specification and elaborated in Figure 48. Note that the creation of LAA carriers is identical to the creation of regular carriers but in the unlicensed bands. O-RU responds to the configuration request with success or failure.

Note also that carrier-selection algorithm at the O-DU (i.e., selecting the “best” channel based on the reported measurements from the O-RU) is an implementation issue and is out of the scope of this document.

Annex A Alarm definition

This section contains example alarms which may be reported.

Obviously, alarms that are not applicable in given HW design or SW configuration shall not be reported. For example, alarms related to fan monitoring are applicable to HW variants with fans.

In many cases alarm detection method is HW specific. It is assumed that alarm detection method is reliable to avoid undetected alarms and false alarms. It is also expected that the NETCONF Server is applying mechanisms to avoid unreasonably fast toggling of alarms’ state.

The example alarms table has following columns:

Fault id – Numerical identifier of alarm. This ID shall be used in <alarm-notif> message (fault-id parameter).

Name – Name of the alarm.

Meaning – Description of alarm, describes high level meaning of the alarm.

Start condition – Defines conditions which must be fulfilled to generate alarm. If filtering time is needed, then it must be defined in this column.

Cancel condition – Defines conditions which must be fulfilled to cancel alarm. If filtering time is needed, then it must be defined in this column.

NETCONF Server actions on detection – Defines actions of the NETCONF Server after alarm has been detected.

NETCONF Server actions on cancel – Defines actions of NETCONF Server after alarm has been cancelled.

System recovery actions – Describes gNB level recovery actions of the NETCONF Client after alarm has been indicated by NETCONF Server. This field is informative only; actions taken by the NETCONF Client are not restricted nor defined in this document. System recovery action "Reset" refers to NETCONF Client forcing reset of O-RU.

Source – Defines possible sources of the alarm (alarm is within O-RU). The following are XML encoding provides examples of the component names defined in various YANG modules that may be alarm sources inside the O-RU.

- **< hardware xmlns= "urn:o-ran:hardware:1.0"><component><o-ran-name/></component></hardware>**
- **<fan-tray xmlns= "urn:o-ran:fan:1.0"><fan-state><name/></fan-state></fan-tray>**
- **<sync xmlns= "urn:o-ran:sync:1.0"><gnss-state><name/></gnss-state></sync>**
- **<external-io xmlns= "urn:o-ran:externalio:1.0”><input><name/></input></external-io>**
- **<external-io xmlns= "urn:o-ran:externalio:1.0”><output><name/></output></external-io>**
- **<ald-ports-io xmlns= "urn:o-ran:ald-port:1.0”><ald-port><name/></ald-port></ald-ports-io>**
- **<port-transceivers xmlns= "urn:o-ran:transceiver:1.0”><port-transceiver-data><name/></ port-transceiver-data ></port-transceivers>**
- **<interfaces xmlns= "urn:ietf:params:xml:ns:yang:ietf-interfaces”><interface><name/></interface></interfaces>**
- **<processing-elements xmlns= "urn:o-ran:processing-element.1.0”><ru-elements><name/></ru-elements></processing-elements>**
- **<-ran-uplane-conf xmlns= "urn:o-ran:uplane-conf.1.0”><low-level-tx-links><name/></low-level-tx-links></ o-ran-uplane-conf>**
- **<o-ran-uplane-conf xmlns= "urn:o-ran:uplane-conf.1.0”><low-level-rx-links><name/></low-level-rx-links></o-ran-uplane-conf>**

Note: If Source will not fit to any of above, or is empty, it means that external device (like Antenna Line Devices) cause an alarm (fault is out of the O-RU). Then additional text in alarm notification is needed to clearly say what may be possible fault source.

Severity – Defines severity of the alarm [30].

Critical – sub unit for which alarm has been generated is not working and cannot be used.

Major – sub unit for which alarm has been generated is degraded, it can be used but performance might be degraded.

Minor – sub unit for which alarm has been generated is still working.

1

Fault id	Name	Meaning	Start condition	Cancel condition	NETCONF Server actions on detection	NETCONF Server actions on cancel	System recovery actions	Source	Severity
1	Unit temperature is high	Unit temperature is higher than expected.	Unit temperature exceeded HW implementation specific value for reasonably long filtering time (e.g. 1 minute).	Unit temperature is below HW implementation specific value for reasonably long filtering time (e.g. 1 minute).	SW implementation specific.	None.	None.	Module	Minor
2	Unit dangerously overheating	Unit temperature is dangerously high.	Unit temperature exceeded HW implementation specific value for reasonably long filtering time (e.g. 1 minute).	Unit temperature is below HW implementation specific value for reasonably long filtering time (e.g. 1 minute). AND Ambient temperature is below predefined HW implementation specific value	Unit deactivates all carriers to prevent HW damage.	None.	None.	Module	Critical
3	Ambient temperature violation	Calculated ambient temperature value goes outside the allowed ambient temperature range.	Calculated ambient temperature goes outside the allowed HW specific ambient temperature range.	Calculated ambient temperature not any more outside the allowed HW specific ambient temperature range	SW implementation specific.	None.	None.	Module	Minor
4	Temperature too low	During startup: The temperature inside the unit is too low. Heating of unit is ongoing. Wait until the alarm is canceled. During runtime: The temperature inside the module is too low.	Unit temperature is below HW implementation specific value.	Unit temperature is x Celsius above HW implementation specific value. Additionally: cancellation of critical alarm (reported during startup) is mandatory within x minutes.	HW implementation specific (e.g. enable heating). Note that actions taken must not interfere with normal unit operation if such is commanded by NETCONF Client	HW implementation specific (e.g. disable the heating). Note that actions taken must not interfere with normal unit operation if such is commanded by NETCONF Client.	None.	Module	Critical during startup Minor during runtime
5	Cooling fan broken	Fan(s) do not run.	HW implementation specific.	HW implementation specific.	None.	None.	None.	Fan supervision	Critical (if cooling is severely degraded) Major (otherwise)

Fault id	Name	Meaning	Start condition	Cancel condition	NETCONF Server actions on detection	NETCONF Server actions on cancel	System recovery actions	Source	Severity
6	No fan detected	Unit cannot identify the used fan type or the fan is not installed at all.	HW implementation specific.	HW implementation specific.	SW implementation specific.	None.	None.	Fan supervision	Minor
7	Tuning failure	A filter has not been able to tune on an appropriate sub-band properly.	HW implementation specific.	HW implementation specific.	None.	None.	None.	Antenna line	Critical
8	Filter unit faulty	Major failure has been detected by the filter.	HW implementation specific.	HW implementation specific.	None.	None.	None.	Antenna line	Critical
9	Transmission quality deteriorated	The TX signal quality may be out of specification limits.	HW and SW implementation specific.	HW and SW implementation specific.	None.	None.	None.	Antenna line	Major
10	RF Module overvoltage protection faulty	Module's overvoltage protection is broken.	HW implementation specific.	None.	None.	None.	None.	Module	Minor
11	Configuring failed	Configuration failed because of a HW or SW fault.	SW or HW fault detected during configuration.	None.	None.	None.	Reset.	Module	Critical
12	Critical file not found	Critical configuration file is missing.	Critical configuration file is detected missing.	None.	None.	None.	Reset.	Module	Critical
13	File not found	Non-critical configuration file is missing.	Non-critical configuration file is detected missing.	None.	None.	None.	None.	Module	Major
14	Configuration file corrupted	conflicting or corrupted configuration data.	conflicting or corrupted configuration data detected.	Unit detects that previously missing file is present.	None.	None.	None.	Module or antenna line	Major
15	Unit out of order	The Unit is out of order because of a software or hardware fault.	HW and SW implementation specific.	HW and SW implementation specific.	None.	None.	Reset.	Module	Critical
16	Unit unidentified	The permanent memory in the module is corrupted and the module product code or serial number is missing, or the module product code is unknown.	Not able to read data from information storage or data is such that module identity or serial number is missing or module identity is unknown.	None.	None.	None.	None.	Module	Major
17	No external sync source	The Unit lost lock to the incoming clock.	HW implementation specific.	HW implementation specific.	None.	None.	Reset.	Module	Major

Fault id	Name	Meaning	Start condition	Cancel condition	NETCONF Server actions on detection	NETCONF Server actions on cancel	System recovery actions	Source	Severity
18	Synchronization Error	Unit is out of synchronization.	HW implementation specific.	HW implementation specific.	None.	None.	Reset or none.	Module or antenna line	Critical
19	TX out of order	TX path is not usable.	HW implementation specific.	HW implementation specific.	None.	None.	Reset or none.	Antenna line	Critical
20	RX out of order	RX path is not usable.	HW implementation specific.	None.	None.	None.	Reset or none.	Antenna line	Critical
21	Increased BER detected on the optical connection	Increased bit error rate has been detected on the optical link which results in sporadical errors in downlink baseband processing.	HW implementation specific (the detected BER on optical link is degrading RF operation).	HW implementation specific (the detected BER on optical link is not degrading RF operation).	Module Agent starts HW implementation specific recovery to keep the RF operation ongoing.	Module Agent stops HW implementation specific recovery actions.	None.	Module	Major
22	Post test failed	Power-on self test failed at start-up.	HW and SW implementation specific.	None.	Unit reset x times for recovery.	None.	None.	Module	Critical
23	FPGA SW update failed	The FPGA software update has failed.	FPGA SW checksum is not correct match after FPGA SW update is detected.	None.	None.	None.	None.	Module	Major
24	Unit blocked	Unit is blocked.	Parameter **blocked** of the Module element is set to "TRUE".	Parameter **blocked** of the Module Element is set to "FALSE"	Blocked unit shuts down all RF emission and turns off power on antenna lines and ALD ports	None.	None.	Module	Critical
25	Reset Requested	Unit detected a transient problem which significantly affects operation that requires a reset as a recovery.	HW implementation specific	None.	None.	None.	Reset.	Module	Critical
26	Power Supply Faulty	Input power to module has fault, unstable or broken.	HW implementation specific	None	None	None	Reset or None	Module	Critical, Major or Minor
27	Power Amplifier faulty	One of power amplifiers in module has fault, unstable or broken	HW implementation specific	None	None	None	Reset or None	Tx-array and/or antenna line	Major
28	C/U-plane logical Connection faulty	One of logical C/U-plane connection has fault, unstable or broken.	One of C/U-plane processing elements detects the error of C/U-plane connection faulty.	SW implementation specific	SW implementation specific (Reconfiguration of C/U-plane logical connection)	None	Reconfiguration of C/U-plane logical connection	Tx-link and/or Rx-link	Major, Minor or warning

Fault id	Name	Meaning	Start condition	Cancel condition	NETCONF Server actions on detection	NETCONF Server actions on cancel	System recovery actions	Source	Severity
29	Transceiver Fault	Unit has detected a transceiver fault	HW and SW implementation specific.	None.	None	None.	None.	Module	Critical
30	Interface Fault	Unit has detected a fault with one of its interfaces	HW and SW implementation specific.	None.	Unit reset x times for recovery.	None.	None.	Module	Major or Critical
31	Unexpected C/U-plane message content fault	C/U-plane message content was faulty for undetermined reason.	C/U-plane detects unexpected message content.	Carrier affected by this fault was removed	SW implementation specific	None	None	Tx-link and/or Rx link	Major, Minor or warning

Annex B Counter definition

measurement-group	measurement-object	report-info	object-unit	Note
transceiver-stats	RX_POWER	**max and time** **min and time** **first and time** **latest and time** **frequency-table**	PORT_NUMBER	Type decimal64 including 4 fraction-digits for **max**, **min**, **first** and **latest**. A parameter date-and-time is reported for each additionally. Configurable parameters: **function**, **bin-count**, **lower-bound**, **upper-bound** are defined. For more detail see B.1.1 and B.1.2. Type uint32 is used for **frequency-table**.
	TX_POWER			
	TX_BIAS_COUNT			
	VOLTAGE			
	TEMPARATURE			
rx-window-stats	RX_ON_TIME	**count**	RU, TRANSPORT, or EAXC_ID	Type yang: counter64 is used for the count. When **object-unit** is EAXC_ID, TRANSPORT is reported as additional parameter for EAXC_ID.
	RX_EARLY			
	RX_LATE			
	RX_CORRUPT			
	RX_DUPL			
	RX_TOTAL			

Table 3: Counters definition

A parameter: **measurement-interval** is defined per group of **measurement-object**s.

A parameter: **active** is defined per **measurement-object**.

The **object-unit** for the **measurement-object** of rx-window-measurement can be selected per RU, per TRANSPORT, or EAXC_ID. RU is assumed to support one of the **object-unit**s for the rx-window-measurement.

TRANSPORT indicates the **name** of **transport-flow** in o-ran-processing-element YANG.

The type Uint16 is used for EAXC_ID. Measurement result shall contain additional information **name** for its **transport-flow** when EAXC_ID is selected for the **object-unit**.

A feature "GRANULARITY-EAXC-ID-MEASUREMENT" and a feature "GRANULARITY-TRANSPORT-MEASUREMENT" are defined as optional definition in O-RU.

B.1 Transceiver Statistics

The transceiver-measurement includes the performance measurement of transceivers as shown in the following table.

Measurement-object-id	measurement-object	Description
11	RX_POWER	Measured Rx input power in mW
12	TX_POWER	Measured Tx input power in mW.

13	TX_BIAS_COUNT	Internally measured Tx Bias Current in mA
14	VOLTAGE	Internally measured transceiver supply voltage in mV
15	TEMPARATURE	Internally measured optional laser temperature in degrees Celsius.

Table 4: Transceiver Measurements

B.1.1 Statistics Calculation

When configured by the NETCONF client, the O-RU captures value of monitored parameters. Then the O-RU calculates x = f(s), where f(s) is a function selected for specific statistics instance. The function f(s) can be one of the following:

- f(s) = s
- f(s) = $LOG_{10}(s)$,

where $LOG_{10}(s)$ is logarithm with base 10. To avoid issues with infinity, the O-RU assumes that for $s < 10^{-128}$ value of $LOG_{10}(s)$ is -128.

The value of x = f(s) is applied to **first**, **latest**, **min** and **max** values; related timestamps are also updated; frequency table is updated as described in the following section.

When local measurement interval, which is not same as **transceiver-measurement-interval** of the **measurement-object**, passes the O-RU captures value of a monitored parameter (s). Then the O-RU calculates x = f(s), where f is a function selected for specific parameter. The local measurement interval is up to the O-RU implementation matter and typically around 10 sec – 60 sec at earliest.

The value of x = f(s) is applied to latest value; related timestamp is updated.
The O-RU updates statistics:

- If x < min value then x is applied to min value and related timestamp is updated.
- If x > max value then x is applied to max value and related timestamp is updated.
- Value of x is used to update frequency table as described in section Frequency Table below.

After updates O-RU waits another interval to elapse.

B.1.2 Frequency Table Generation

Let n = **bin-count**, a = **lower-bound**, b = **upper-bound**, x = f(s) where s is value of monitored parameter and f is a function selected for statistics via parameter **function**.

- If n = 0 then frequency table is empty and is not updated.
- If n > 0 there are n bins: h_k where k = 0...n-1. Initial value of each bin is zero ($h_k = 0$ for k = 0...n-1).
- If x < a then bin h_0 is incremented.
- If $b \le x$ and n > 1 then bin h_{n-1} is incremented.
- If $a \le x$ and x < b and n > 2 then bin h_k is incremented for k such that
- $k-1 \le (n-2) * (x-a) / (b-a) < k$. 32

Note value of bin should saturate at maximum without overflowing (the value is not incremented above 2^{32-1}).
Equivalently:

- For k = 0, h_k is a number of values x such that x < a.
- For k = 1 ... n-2, h_k is a number of values x such that
- $a + (b-a) * (k-1) / (n-2) \le x < a + (b-a) * (k) / (n-2)$.
- For k = n-1, h_k is a number of values x such that $b \le x$.

Example:

function = LOG_{10}, **bin-count** = 14, **lower-bound** = -12, **upper-bound** = 0

- parameter value s = 0, x = f(0) = -128, -128 < -12 = a → h_0 is incremented
- parameter value s = $1e^{-12}$, x = f($1e^{-12}$) = -12, (14-2)*(-12-(-12))/(0-(-12)) = 12*0/12 < 1 → h_1 is incremented

- parameter value s = $9.99e^{-12}$, x = $f(9.99e^{-12})$ = -11.0004, (14-2)*(-11.0004-(-12))/(0-(-12)) = 12*0.0004/12 < 1 → h_1 is incremented
- parameter value s = $1e^{-1}$, x = $f(1e^{-1})$ = -1, (14-2)*(-1-(-12))/(0-(-12)) = 12*11/12 < 12 → h_{12} is incremented
- parameter value s = 1, x = f(1) = 0, b = 0 ≤ 1 → h_{13} is incremented

B.2 Rx Window Statistics

The rx-window-measurement includes the performance measurement for the reception window as following table.

Measurement-object-id	measurement-object	Description
1	RX_ON_TIME	The number of data packet received on time (applies to user data reception window) within the **rx-window-measurement-interval**
2	RX_EARLY	The number of data packet received too early (applies to user data reception window) within the **rx-window-measurement-interval**
3	RX_LATE	The number of data packet received too late (applies to user data reception window) within the **rx-window-measurement-interval**
4	RX_CORRUPT	The number of data packet, which is corrupt or whose header is incorrect, received within the **rx-window-measurement-interval**
5	RX_DUPL	The number of data packet, which is duplicated with other data packet, received within the **rx-window-measurement-interval**
6	RX_TOTAL	The total number of received data packet, within the **rx-window-measurement-interval**

Table 5: Rx Window Measurement

Annex C Optional Multi-Vendor Functionality

C.1: Optional Namespace

Some of the YANG models are optional for the O-RU to support. In this version of the management plane specification, the following YANG models are optional to support. If an O-RU/NETCONF server does not return the namespace associated with an optional YANG model, the NETCONF client can infer that the O-RU does not support the optional capability associated with the model.

No	Optional Functionality	Reference	Namespace
1	Antenna Line Device	Chapter 11.3	"urn:o-ran:ald-port:x.y" "urn:o-ran:ald: x.y "
2	External IO Port	Chapter 11.4	"urn:o-ran:external-io:x.y "
3	eCPRI delay measurement	Chapter 4.7	"urn:o-ran:message5:x.y "
4	UDP Echo functionality for IP based transport verification	Chapter 4.6	"urn:o-ran:udpecho:x.y "

5	Beamforming	Chapter 12.4	"urn:o-ran:beamforming:x.y "
6	FAN	-	“urn:o-ran:fan:x.y”
7	LAA	Chapter 13	"urn:o-ran:laa:x.y " "urn:o-ran:laa-operations:x.y "

Table 6: Optional O-RAN Namespace

C.2: Optional YANG Features

Some of the O-RAN defined YANG models define optional feature support. The optional multi-vendor features defined in the O-RAN defined YANG models are shown below.

No	Optional Feature	Namespace	Feature name
1	Adaptive O-RU delay profile	"urn:o-ran:delay:x.y "	ADAPTIVE-RU-PROFILE
2	O-RU Energy saving	"urn:o-ran:hardware:x.y "	ENERGYSAVING
3	Alias MAC address based C/U transport	"urn:o-ran:interfaces:x.y "	ALIASMAC-BASED-CU-PLANE
4	UDP/IP based C/U Transport	"urn:o-ran:interfaces:x.y "	UDPIP-BASED-CU-PLANE
5	Dynamic Beamforming Configuration	"urn:o-ran:beamforming:x.y "	MODIFY-BF-CONFIG
6	GNSS Support	"urn:o-ran:sync:x.y "	GNSS
7	ALD overcurrent reporting	"urn:o-ran:ald-port:x.y "	OVERCURRENT-SUPPORTED
8	TRANSPORT in rx-window-measurement	"urn:o-ran:performance-management:x.y "	GRANULARITY-TRANSPORT-MEASUREMENT
9	EAXC_ID in rx-window-measurement	"urn:o-ran:performance-management:x.y "	GRANULARITY-EAXC-ID-MEASUREMENT
10	LAA Support	"urn:o-ran: module-cap "	LAA
11	Transport Fragmentation	"urn:o-ran:module-cap:x.y "	TRANSPORT-FRAGMENTATION
12	GNSS Anti Jamming	"urn:o-ran:sync:x.y "	ANTI-JAM

Table 7: Optional O-RAN WG4 defined feature support

Some of the O-RAN defined YANG models augment existing YANG models which have optional features defined. The optional features defined in these “common” models are shown in the table below.

No	Optional Feature	Namespace	Feature name
1	RFC 6933: Entity MIB	"urn:ietf:params:xml:ns:yang:ietf-hardware"	entity-mib
2	RFC 4268: Entity State MIB	"urn:ietf:params:xml:ns:yang:ietf-hardware"	hardware-state
3	RFC 3433: Entity Sensor Management Information Base	"urn:ietf:params:xml:ns:yang:ietf-hardware"	hardware-sensor
4	O-RU allows user-controlled interfaces to be named arbitrarily	"urn:ietf:params:xml:ns:yang:ietf-interfaces"	arbitrary-names
5	O-RU supports pre-provisioning of interface configuration, i.e., it is possible to configure an interface whose physical interface hardware is not present on the device	"urn:ietf:params:xml:ns:yang:ietf-interfaces"	pre-provisioning
6	RFC 2863: The Interfaces Group MIB	"urn:ietf:params:xml:ns:yang:ietf-interfaces"	if-mib
7	O-RU supports configuring non-contiguous subnet masks	"urn:ietf:params:xml:ns:yang:ietf-ip"	ipv4-non-contiguous-netmasks
8	O-RU supports privacy extensions for stateless address autoconfiguration in IPv6	"urn:ietf:params:xml:ns:yang:ietf-ip"	ipv6-privacy-autoconf

Table 8: Optional feature support in common models

C.3: Optional Capabilities Exposed Using O-RAN YANG Models

In addition to optional namespaces and optional features within supported namespaces, certain O-RAN defined YANG models are used to be able to expose support for certain optional capabilities by the O-RU.

No	Optional Feature	Namespace	Leaf
1	Type of synchronization source supported by O-RU	"urn:o-ran:sync:x.y "	/ sync/:sync-status/supported-reference-types
2	O-RU supports extended Category A operation – more than 8 spatial streams	"urn:o-ran:module-cap:x.y "	/module-capability/ru-capabilities/ ru-supported-category and /module-capability/ru-capabilities/ number-of-spatial-streams
3	O-RU supports Category B operation – precoding in the O-RU	"urn:o-ran:module-cap:x.y "	/module-capability/ru-capabilities/ ru-supported-category
4	O-RU supports the capability to apply the modified beamforming configuration by using rpc activate-beamforming-config without deletion of tx-array-carriers and rx-array-carriers	"urn:o-ran:beamforming:x.y "	/beamforming-configuration/ operational-properties/update-bf-non-delete
5	O-RU supports the capability to store the modified beamforming configuration file in the reset persistent memory	"urn:o-ran:beamforming:x.y "	/beamforming-configuration/ operational-properties/persistent-bf-files
6	Optional VLAN optimized searching	"urn:o-ran:mplane-interfaces:x.y "	/mplane-info/searchable-mplane-access-vlans-info
7	Configurable CoS marking for C, U and M-Plane	"urn:o-ran:interfaces:x.y "	augmented /if:interfaces/if:interface: with u-plane-marking c-plane-marking and m-plane-marking
8	Configurable DSCP marking for C, U and M-Plane	"urn:o-ran:interfaces:x.y "	augmented /if:interfaces/if:interface: with u-plane-marking c-plane-marking and m-plane-marking
9	Ethernet Frame MTU	"urn:o-ran:interfaces:x.y”	augmented /if:interfaces/if:interface: with l2-mtu
10	VLAN Tagging	"urn:o-ran:interfaces:x.y”	augmented /if:interfaces/if:interface: with vlan-tagging
11	IEEE 1914.3 header support	"urn:o-ran:operations:x.y "	/operational-info/o-ran-split/optional-header-support
12	eCPRI Concatenation support	"urn:o-ran:operations:x.y "	/operational-info/o-ran-split/ecpri-concatenation-support
13	O-RU local management of the LAA contention window	"urn:o-ran:module-cap:x.y "	/module-capability/band-capabilities/sub-band-info/self-configure
14	O-RU supports LAA ending in Downlink Pilot Time Slot (DwPTS)	"urn:o-ran:laa:x.y”	/laa-config/ laa-ending-dwpts-supported

Table 9: Optional capabilities in O-RAN defined YANG models

C.4: Optional Capabilities Exposed Using Common YANG Models

When O-RAN defines augmentation of existing YANG models, these models may expose support for certain optional capabilities by the O-RU.

No	Optional Feature	Namespace	Leaf
1	IPv6 Supported by O-RU	"urn:ietf:params:xml:ns:yang:ietf-ip"	augmented /if:interfaces/if:interface: with ipv6

Table 10: Optional capabilities in common YANG models

Annex D YANG Module Graphical Representation

The different released version of the set of YANG modules for the O-RU can be downloaded from O-RAN's website http://www.o-ran.org/resources/. The YANG models are available in a zip file, whose name is used to represent the version of the YANG model and follows the numerical format defined in subsection 1.1 with the periods replaced with "-", i.e., YANG models for release 1.0.0 of the M-Plane specification are availabile in the file 1-0-0.zip. This zip file includes all published revisions of the YANG models supporting a particular release of the M-Plane specification.

This Annex provides a set of "tree-views" of the modules to provide a simplified graphical representation of the data models. These trees have been automatically generated using the pyang YANG validator tool. If there are any inconsistencies between the tree representation in this Annex and the yang models, the yang models shall take precedence.

D.1 System Folder

D.1.1 o-ran-supervision.yang Module

The format for the supervision module is provided below.

```
module: o-ran-supervision

  rpcs:
    +---x supervision-watchdog-reset
       +---w input
       |  +---w supervision-notification-interval?   uint16
       |  +---w guard-timer-overhead?                uint16
       +--ro output
          +--ro next-update-at?   yang:date-and-time]

  notifications:
    +---n supervision-notification
```

D.1.2 o-ran-usermgmt.yang Module

The format for the user management module is provided below.

```
module: o-ran-usermgmt
  +--rw user-profile
     +--rw user* [name]
        +--rw name         username-type
        +--rw password?    password-type
        +--rw enabled?     boolean


  rpcs:
    +---x chg-password
       +---w input
       |  +---w currentPassword      password-type
       |  +---w newPassword          password-type
       |  +---w newPasswordConfirm   password-type
       +--ro output
          +--ro status           enumeration
          +--ro status-message?   string
```

D.1.3 o-ran-hardware.yang Module

The format for the hardware module is provided below.

```
module: o-ran-hardware
  augment /hw:hardware/hw:component:
    +--ro label-content
    | +--ro model-name?      boolean
    | +--ro serial-number?   boolean
    +--ro product-code             string
    +--rw energy-saving-enabled?   boolean {ENERGYSAVING}?
  augment /hw:hardware/hw:component:
    +--rw o-ran-name    -> /hw:hardware/component/name
  augment /hw:hardware/hw:component/hw:state:
    +--ro power-state?          energysaving-state {ENERGYSAVING}?
    +--ro availability-state?   availability-type
  augment /hw:hardware-state-oper-enabled:
    +--ro availability-state?   -> /hw:hardware/component/state/o-ran-hw:availability-state
  augment /hw:hardware-state-oper-disabled:
    +--ro availability-state?   -> /hw:hardware/component/state/o-ran-hw:availability-state
```

D.1.4 o-ran-fan.yang Module

The format for the fan module is provided below.

```
module: o-ran-fan
  +--ro fan-tray
     +--ro fan-state* [name]
        +--ro name                    string
        +--ro fan-location?           uint8
        +--ro present-and-operating   boolean
        +--ro vendor-code?            uint8
        +--ro fan-speed?              percent
        +--ro target-speed?           uint16
```

D.1.5 o-ran-fm.yang Module

The format for the fault management module is provided below.

```
module: o-ran-fm
  +--ro active-alarm-list
     +--ro active-alarms* []
        +--ro fault-id           uint16
        +--ro fault-source       string
        +--ro affected-objects* []
        | +--ro name    string
        +--ro fault-severity     enumeration
        +--ro is-cleared         boolean
        +--ro fault-text?        string
        +--ro event-time         yang:date-and-time

  notifications:
    +---n alarm-notif
       +--ro fault-id           uint16
       +--ro fault-source       string
       +--ro affected-objects* []
       | +--ro name    string
       +--ro fault-severity     enumeration
       +--ro is-cleared         boolean
       +--ro fault-text?        string
       +--ro event-time         yang:date-and-time
```

D.2 Operations Folder

D.2.1 o-ran-operations.yang Module

The format for the operations module is provided below.

```
module: o-ran-operations
  +--rw operational-info
     +--ro declarations
     | +--ro supported-mplane-version?     version
     | +--ro supported-cusplane-version?   version
     | +--ro supported-header-mechanism* [protocol]
     |    +--ro protocol                      enumeration
     |    +--ro ecpri-concatenation-support?   boolean
     |    +--ro protocol-version?              version
     +--ro operational-state
     | +--ro restart-cause?      enumeration
     | +--ro restart-datetime?   yang:date-and-time
```

```
   +--rw clock
   |  +--rw timezone-utc-offset?   int16
   +--rw re-call-home-no-ssh-timer?   uint16

 rpcs:
  +---x reset
```

D.2.2 o-ran-file-management.yang Module

The format for the file management module is provided below

```
module: o-ran-file-management

 rpcs:
  +---x file-upload
  |  +---w input
  |  |  +---w local-logical-file-path    string
  |  |  +---w remote-file-path           string
  |  |  +---w (credentials)?
  |  |     +--:(password)
  |  |     |  +---w password!
  |  |     |     +---w password    string
  |  |     +--:(certificate)
  |  |        +---w certificate!
  |  +--ro output
  |     +--ro status?          enumeration
  |     +--ro reject-reason?   string
  +---x retrieve-file-list
  |  +---w input
  |  |  +---w logical-path        string
  |  |  +---w file-name-filter?   string
  |  +--ro output
  |     +--ro status?          enumeration
  |     +--ro reject-reason?   string
  |     +--ro file-list*       string
  +---x file-download
     +---w input
     |  +---w local-logical-file-path    string
     |  +---w remote-file-path           string
     |  +---w (credentials)?
     |     +--:(password)
     |     |  +---w password!
     |     |     +---w password    string
     |     +--:(certificate)
     |        +---w certificate!
     +--ro output
        +--ro status?          enumeration
        +--ro reject-reason?   string

 notifications:
  +---n file-upload-notification
  |  +--ro local-logical-file-path    string
  |  +--ro remote-file-path           string
  |  +--ro status?                    enumeration
  |  +--ro reject-reason?             string
  +---n file-download-event
     +--ro local-logical-file-path    string
     +--ro remote-file-path           string
     +--ro status?                    enumeration
     +--ro reject-reason?             string
```

D.2.3 o-ran-software-management.yang Module

The format for the software management module is provided below

```
module: o-ran-software-management
  +--ro software-inventory
     +--ro software-slot* [name]
        +--ro name             string
        +--ro status           enumeration
        +--ro active?          boolean
        +--ro running?         boolean
        +--ro access?          enumeration
        +--ro product-code?    -> /hw:hardware/component/o-ran-hw:product-code
        +--ro vendor-code?     string
        +--ro build-id?        string
```

```
       +--ro build-name?      string
       +--ro build-version?   string
       +--ro files* [name]
          +--ro name          string
          +--ro version?      string
          +--ro local-path    string
          +--ro integrity?    enumeration

 rpcs:
   +---x software-download
   |  +---w input
   |  |  +---w remote-file-path   inet:uri
   |  |  +---w (credentials)?
   |  |     +--:(password)
   |  |     |  +---w password!
   |  |     |     +---w password    string
   |  |     +--:(certificate)
   |  |        +---w certificate!
   |  +--ro output
   |     +--ro status                 enumeration
   |     +--ro error-message?         string
   |     +--ro notification-timeout?  int32
   +---x software-install
   |  +---w input
   |  |  +---w slot-name     -> /software-inventory/software-slot/name
   |  |  +---w file-names*   string
   |  +--ro output
   |     +--ro status          enumeration
   |     +--ro error-message?  string
   +---x software-activate
      +---w input
      |  +---w slot-name    -> /software-inventory/software-slot/name
      +--ro output
         +--ro status                 enumeration
         +--ro error-message?         string
         +--ro notification-timeout?  int32

 notifications:
   +---n download-event
   |  +--ro file-name        string
   |  +--ro status?          enumeration
   |  +--ro error-message?   string
   +---n install-event
   |  +--ro slot-name?       -> /software-inventory/software-slot/name
   |  +--ro status?          enumeration
   |  +--ro error-message?   string
   +---n activation-event
      +--ro slot-name?       -> /software-inventory/software-slot/name
      +--ro status?          enumeration
      +--ro return-code?     uint8
      +--ro error-message?   string
```

D.2.4 o-ran-lbm.yang Module

The format for the (Ethernet) loobback module is provided below

```
module: o-ran-lbm
  +--rw md-data-definitions
     +--rw maintenance-domain* [id]
        +--rw id                          string
        +--rw name?                       string
        +--rw md-level?                   md-level-type
        +--rw maintenance-association* [id]
           +--rw id                 string
           +--rw name?              string
           +--rw component-list* [component-id]
              +--rw component-id                          uint32
              +--rw name?                                 string
              +--rw vid*                                  -> /if:interfaces/interface/o-ran-int:vlan-id
              +--rw remote-meps*                          mep-id-type
              +--rw maintenance-association-end-point* [mep-identifier]
                 +--rw mep-identifier          mep-id-type
                 +--rw interface               -> /if:interfaces/interface/name
                 +--rw primary-vid             -> /if:interfaces/interface/o-ran-int:vlan-id
                 +--rw administrative-state    boolean
                 +--ro mac-address?            -> /if:interfaces/interface/o-ran-int:mac-address
              +--ro loopback
                 +--ro replies-transmitted    yang:counter32
```

D.2.5 -ran-udp-echo.yang Module

The format for the udp echo module is provided below

```
module: o-ran-udp-echo
  +--rw udp-echo {o-ran-int:UDPIP-BASED-CU-PLANE}?     +--rw enable-udp-echo?           boolean
     +--rw dscp-config?                 enumeration
     +--ro echo-replies-transmitted?   uint32
```

D.2.6 o-ran-ecpri-delay.yang Module

The format for the ecpri delay management module is provided below

```
module: o-ran-ecpri-delay
  +--rw ecpri-delay-message
     +--ro ru-compensation
     |  +--ro tcv2?   uint32
     |  +--ro tcv1?   uint32
     +--rw enable-message5?     boolean
     +--rw message5-sessions
        +--rw session-parameters* [session-id]
           +--rw session-id                 uint32
           +--rw processing-element-name?   -> /element:processing-elements/ru-elements/name
           +--ro flow-state
              +--ro responses-transmitted?   uint32
              +--ro requests-transmitted?    uint32
              +--ro followups-transmitted?   uint32
```

D.2.7 o-ran-performance-management.yang Module

The format for the performance management module is provided below

```
module: o-ran-performance-management
  +--rw performance-measurement-objects
     +--rw enable-SFTP-upload?                 boolean
     +--rw enable-random-file-upload?          boolean
     +--rw remote-SFTP-uploads* [remote-SFTP-upload-path]
     |  +--rw remote-SFTP-upload-path    inet:uri
     |  +--rw (credentials)?
     |     +--:(password)
     |     |  +--rw password!
     |     |     +--rw password    string
     |     +--:(certificate)
     |        +--rw certificate!
     +--rw transceiver-measurement-interval?   uint16
     +--rw rx-window-measurement-interval?     uint16
     +--rw notification-interval?              uint16
     +--rw file-upload-interval?               uint16
     +--ro max-bin-count                       uint16
     +--rw transceiver-measurement-objects* [measurement-object]
     |  +--rw measurement-object               enumeration
     |  +--rw active?                          boolean
     |  +--rw report-info*                     enumeration
     |  +--rw object-unit                      enumeration
     |  +--rw function?                        enumeration
     |  +--rw bin-count?                       Uint16
     |  +--rw lower-bound?                     decimal64
     |  +--rw upper-bound?                     decimal64
     |  +--ro transceiver-measurement-result* [object-unit-id]
     |     +--ro object-unit-id    -> /if:interfaces/interface/o-ran-int:port-reference/o-ran-port-number
     |     +--ro min
     |     |  +--ro value?   decimal64
     |     |  +--ro time?    yang-types:date-and-time
     |     +--ro max
     |     |  +--ro value?   decimal64
     |     |  +--ro time?    yang-types:date-and-time
     |     +--ro first
     |     |  +--ro value?   decimal64
     |     |  +--ro time?    yang-types:date-and-time
     |     +--ro latest
     |     |  +--ro value?   decimal64
     |     |  +--ro time?    yang-types:date-and-time
```

```
   |     +--ro frequeny-table*   uint32
   +--rw rx-window-measurement-objects* [measurement-object]
      +--rw measurement-object            enumeration
      +--rw active?                       boolean
      +--rw object-unit?                  enumeration
      +--rw report-info?                  enumeration
      +--ro (object-unit-id)?
         +--:(RU)
         |  +--ro name?                   -> /hw:hardware/component/name
         |  +--ro count                   uint64
         +--:(TRANSPORT)
         |  +--ro tr-measured-result* []
         |     +--ro name?    -> /o-ran-elements:processing-elements/ru-elements/name
         |     +--ro count    uint64
         +--:(EAXC_ID)
            +--ro eaxc-measured-result* []
               +--ro eaxc-id?         uint16
               +--ro count            uint64
               +--ro transport-name?   -> /o-ran-elements:processing-elements/ru-elements/name

 notifications:
  +---n measurement-result-stats
     +--ro transceiver-stats* [measurement-object]
     | +--ro measurement-object                -> /performance-measurement-objects/transceiver-measurement-objects/measurement-
object
     | +--ro start-time?                      yang-types:date-and-time
     | +--ro end-time?                        yang-types:date-and-time
     | +--ro transceiver-measurement-result* [object-unit-id]
     |    +--ro object-unit-id    -> /if:interfaces/interface/o-ran-int:port-reference/port-number
     |    +--ro min
     |    | +--ro value?   decimal64
     |    | +--ro time?    yang-types:date-and-time
     |    +--ro max
     |    | +--ro value?   decimal64
     |    | +--ro time?    yang-types:date-and-time
     |    +--ro first
     |    | +--ro value?   decimal64
     |    | +--ro time?    yang-types:date-and-time
     |    +--ro latest
     |    | +--ro value?   decimal64
     |    | +--ro time?    yang-types:date-and-time
     |    +--ro frequeny-table*   uint32
     +--ro rx-window-stats* [measurement-object]
        +--ro measurement-object            -> /performance-measurement-objects/rx-window-measurement-objects/measurement-object
        +--ro start-time?                   yang-types:date-and-time
        +--ro end-time?                     yang-types:date-and-time
        +--ro (object-unit-id)?
           +--:(RU)
           | +--ro name?                   -> /hw:hardware/component/name
           | +--ro count                   uint64
           +--:(TRANSPORT)
           | +--ro tr-measured-result* []
           |    +--ro name?    -> /o-ran-elements:processing-elements/ru-elements/name
           |    +--ro count    uint64
           +--:(EAXC_ID)
              +--ro eaxc-measured-result* []
                 +--ro eaxc-id?         uint16
                 +--ro count            uint64
                 +--ro transport-name?   -> /o-ran-elements:processing-elements/ru-elements/name
```

D.2.8 o-ran-uplane-conf.yang Module

The format for the userplane configuration module is provided below

```
module: o-ran-uplane-conf
  +--rw user-plane-configuration
     +--rw low-level-tx-links* [name]
     | +--rw name                        string
     | +--rw processing-element       -> /o-ran-pe:processing-elements/ru-elements/name
     | +--rw tx-array-carrier         -> /user-plane-configuration/tx-array-carriers/name
     | +--rw low-level-tx-endpoint   -> /user-plane-configuration/low-level-tx-endpoints/name
     +--rw low-level-rx-links* [name]
     | +--rw name                        string
     | +--rw processing-element          -> /o-ran-pe:processing-elements/ru-elements/name
     | +--rw rx-array-carrier            -> /user-plane-configuration/rx-array-carriers/name
     | +--rw low-level-rx-endpoint       -> /user-plane-configuration/low-level-tx-endpoints/name
```

```
   |  +--rw user-plane-uplink-marking?   -> /o-ran-pe:processing-elements/enhanced-uplane-mapping/uplane-mapping/up-marking-name
   +--ro endpoint-types* [id]
   |  +--ro id                                   uint16
   |  +--ro supported-section-types* [section-type]
   |  |  +--ro section-type                      uint8
   |  |  +--ro supported-section-extensions*   uint8
   |  +--ro supported-frame-structures*            uint8
   |  +--ro managed-delay-support?                 enumeration
   |  +--ro max-numerology-change-duration?        uint16
   |  +--ro max-control-sections-per-data-section?  uint8
   |  +--ro max-sections-per-symbol?               uint16
   |  +--ro max-sections-per-slot?                 uint16
   |  +--ro max-beams-per-symbol?                  uint16
   |  +--ro max-beams-per-slot?                    uint16
   |  +--ro max-prb-per-symbol?                    uint16
   |  +--ro prb-capacity-allocation-granularity*   uint16
   |  +--ro max-numerologies-per-symbol?           uint16
   +--ro endpoint-capacity-sharing-groups* [id]
   |  +--ro id                                   uint16
   |  +--ro max-control-sections-per-data-section?  uint8
   |  +--ro max-sections-per-symbol?               uint16
   |  +--ro max-sections-per-slot?                 uint16
   |  +--ro max-beams-per-symbol?                  uint16
   |  +--ro max-beams-per-slot?                    uint16
   |  +--ro max-prb-per-symbol?                    uint16
   |  +--ro max-numerologies-per-symbol?           uint16
   |  +--ro max-endpoints?                         uint16
   |  +--ro max-managed-delay-endpoints?           uint16
   |  +--ro max-non-managed-delay-endpoints?       uint16
   +--ro static-low-level-tx-endpoints* [name]
   |  +--ro name                       string
   |  +--ro restricted-interfaces*     -> /if:interfaces/interface/name
   |  +--ro array                      -> /user-plane-configuration/tx-arrays/name
   |  +--ro endpoint-type?             -> ../../endpoint-types/id
   |  +--ro capacity-sharing-groups*   -> ../../endpoint-capacity-sharing-groups/id
   +--ro static-low-level-rx-endpoints* [name]
   |  +--ro name                       string
   |  +--ro restricted-interfaces*     -> /if:interfaces/interface/name
   |  +--ro array                      -> /user-plane-configuration/rx-arrays/name
   |  +--ro endpoint-type?             -> ../../endpoint-types/id
   |  +--ro capacity-sharing-groups*   -> ../../endpoint-capacity-sharing-groups/id
   +--rw low-level-tx-endpoints* [name]
   |  +--rw name                                -> /user-plane-configuration/static-low-level-tx-endpoints/name
   |  +--rw compression!
   |  |  +--rw compression-type                       enumeration
   |  |  +--rw bitwidth?                          uint8
   |  |  +--rw (compression-format)?
   |  |     +--:(no-compresison)
   |  |     +--:(block-floating-point)
   |  |     |  +--rw exponent?                      uint8
   |  |     +--:(block-scaling)
   |  |     |  +--rw block-scalar?                  uint8
   |  |     +--:(u-law)
   |  |     |  +--rw comp-bit-width?                 uint8
   |  |     |  +--rw comp-shift?                     uint8
   |  |     +--:(beam-space-compression)
   |  |     |  +--rw active-beam-space-coeficient-mask*   uint8
   |  |     |  +--rw block-scaler?                  uint8
   |  |     +--:(modulation-compression)
   |  |        +--rw csf?                          uint8
   |  |        +--rw mod-comp-scaler?                   uint16
   |  +--rw frame-structure?                   uint8
   |  +--rw cp-type?                           enumeration
   |  +--rw cp-length                          uint16
   |  +--rw cp-length-other                       uint16
   |  +--rw offset-to-absolute-frequency-center    int32
   |  +--rw number-of-prb-per-scs* [scs]
   |  |  +--rw scs              mcap:scs-config-type
   |  |  +--rw number-of-prb    uint16
   |  +--rw e-axcid
   |     +--rw du-port-bitmask        uint16
   |     +--rw band-sector-bitmask    uint16
   |     +--rw ccid-bitmask           uint16
   |     +--rw ru-port-bitmask        uint16
   |     +--rw eaxc-id                uint16
   +--rw low-level-rx-endpoints* [name]
   |  +--rw name                                -> /user-plane-configuration/static-low-level-rx-endpoints/name
```

```
     | +--rw compression
     | | +--rw compression-type                         enumeration
     | | +--rw bitwidth?                         uint8
     | | +--rw (compression-format)?
     | |    +--:(no-compresison)
     | |    +--:(block-floating-point)
     | |    | +--rw exponent?                         uint8
     | |    +--:(block-scaling)
     | |    | +--rw block-scalar?                     uint8
     | |    +--:(u-law)
     | |    | +--rw comp-bit-width?                    uint8
     | |    | +--rw comp-shift?                        uint8
     | |    +--:(beam-space-compression)
     | |    | +--rw active-beam-space-coeficient-mask*   uint8
     | |    | +--rw block-scaler?                     uint8
     | |    +--:(modulation-compression)
     | |       +--rw csf?                        uint8
     | |       +--rw mod-comp-scaler?                   uint16
     | +--rw frame-structure?                    uint8
     | +--rw cp-type?                         enumeration
     | +--rw cp-length                        uint16
     | +--rw cp-length-other                     uint16
     | +--rw offset-to-absolute-frequency-center    int32
     | +--rw number-of-prb-per-scs* [scs]
     | | +--rw scs             mcap:scs-config-type
     | | +--rw number-of-prb    uint16
     | +--rw ul-fft-sampling-offsets* [scs]
     | | +--rw scs                    mcap:scs-config-type
     | | +--rw ul-fft-sampling-offset?   uint16
     | +--rw e-axcid
     | | +--rw du-port-bitmask       uint16
     | | +--rw band-sector-bitmask    uint16
     | | +--rw ccid-bitmask          uint16
     | | +--rw ru-port-bitmask       uint16
     | | +--rw eaxc-id              uint16
     | +--rw non-time-managed-delay-enabled?       boolean
     +--rw tx-array-carriers* [name]
     | +--rw name                     string
     | +--rw absolute-frequency-center      uint32
     | +--rw center-of-channel-bandwidth    uint64
     | +--rw channel-bandwidth              uint64
     | +--rw active?                     enumeration
     | +--ro state                      enumeration
     | +--ro type                       enumeration
     | +--ro duplex-scheme?                 enumeration
     | +--rw band-number?                  uint16 {mcap:LAA}?
     | +--rw lte-tdd-frame
     | | +--rw subframe-assignment        enumeration
     | | +--rw special-subframe-pattern    enumeration
     | +--rw laa-carrier-configuration {mcap:LAA}?
     | | +--rw ed-threshold-pdsch?          int8
     | | +--rw ed-threshold-drs?           int8
     | | +--rw tx-antenna-ports?           uint8
     | | +--rw transmission-power-for-drs?   int8
     | | +--rw dmtc-period?                enumeration
     | | +--rw dmtc-offset?              uint8
     | | +--rw lbt-timer?               uint16
     | | +--rw max-cw-usage-counter* [priority]
     | |    +--rw priority        enumeration
     | |    +--rw counter-value?   uint8
     | +--rw gain                       decimal64
     | +--rw downlink-radio-frame-offset    uint32
     | +--rw downlink-sfn-offset          int16
     +--rw rx-array-carriers* [name]
     | +--rw name                     string
     | +--rw absolute-frequency-center      uint32
     | +--rw center-of-channel-bandwidth    uint64
     | +--rw channel-bandwidth              uint64
     | +--rw active?                     enumeration
     | +--ro state                      enumeration
     | +--ro type                       enumeration
     | +--ro duplex-scheme?                 enumeration
     | +--rw downlink-radio-frame-offset    uint32
     | +--rw downlink-sfn-offset          int16
     | +--rw gain-correction             decimal64
     | +--rw n-ta-offset               uint32
     +--ro tx-arrays* [name]
```

```
    | +--ro name                              string
    | +--ro number-of-rows                    uint16
    | +--ro number-of-columns                 uint16
    | +--ro number-of-array-layers            uint8
    | +--ro horizontal-spacing?               decimal64
    | +--ro vertical-spacing?                 decimal64
    | +--ro normal-vector-direction
    | | +--ro azimuth-angle?   decimal64
    | | +--ro zenith-angle?    decimal64
    | +--ro leftmost-bottom-array-element-position
    | | +--ro x?   decimal64
    | | +--ro y?   decimal64
    | | +--ro z?   decimal64
    | +--ro polarisations* [p]
    | | +--ro p              uint8
    | | +--ro polarisation    polarisation_type
    | +--ro band-number                        -> /mcap:module-capability/band-capabilities/band-number
    | +--ro max-gain                          decimal64
    | +--ro independent-power-budget                boolean
    +--ro rx-arrays* [name]
    | +--ro name                              string
    | +--ro number-of-rows                    uint16
    | +--ro number-of-columns                 uint16
    | +--ro number-of-array-layers            uint8
    | +--ro horizontal-spacing?               decimal64
    | +--ro vertical-spacing?                 decimal64
    | +--ro normal-vector-direction
    | | +--ro azimuth-angle?   decimal64
    | | +--ro zenith-angle?    decimal64
    | +--ro leftmost-bottom-array-element-position
    | | +--ro x?   decimal64
    | | +--ro y?   decimal64
    | | +--ro z?   decimal64
    | +--ro polarisations* [p]
    | | +--ro p              uint8
    | | +--ro polarisation    polarisation_type
    | +--ro band-number                        -> /mcap:module-capability/band-capabilities/band-number
    | +--ro gain-correction-range
    |    +--ro max    decimal64
    |    +--ro min    decimal64
    +--ro relations* [entity]
      +--ro entity    uint16
      +--ro array1
      | +--ro (antenna-type)?
      |    +--:(tx)
      |    | +--ro tx-array-name?   -> /user-plane-configuration/tx-arrays/name
      |    +--:(rx)
      |       +--ro rx-array-name?   -> /user-plane-configuration/rx-arrays/name
      +--ro array2
      | +--ro (antenna-type)?
      |    +--:(tx)
      |    | +--ro tx-array-name?   -> /user-plane-configuration/tx-arrays/name
      |    +--:(rx)
      |       +--ro rx-array-name?   -> /user-plane-configuration/rx-arrays/name
      +--ro types* [relation-type]
         +--ro relation-type    enumeration
         +--ro pairs* [element-array1]
            +--ro element-array1    uint16
            +--ro element-array2?   uint16

  notifications:
   +---n tx-array-carriers-state-change
   | +--ro tx-array-carriers* [name]
   |    +--ro name     -> /user-plane-configuration/tx-array-carriers/name
   |    +--ro state?   -> /user-plane-configuration/tx-array-carriers/state
   +---n rx-array-carriers-state-change
      +--ro rx-array-carriers* [name]
         +--ro name     -> /user-plane-configuration/rx-array-carriers/name
         +--ro state?   -> /user-plane-configuration/rx-array-carriers/state
```

D.2.9 o-ran-ald Module

The format for the ald module is provided below

```
module: o-ran-ald
  rpcs:
    +---x ald-communication
       +---w input
       |  +---w port-id        -> /ap:ald-ports-io/ald-port/port-id
       |  +---w ald-req-msg?   binary
       +--ro output
          +--ro port-id                      -> /ap:ald-ports-io/ald-port/port-id
          +--ro status                       enumeration
          +--ro error-message?               string
          +--ro ald-resp-msg?                binary
          +--ro frames-with-wrong-crc?       uint32
          +--ro frames-without-stop-flag?    uint32
          +--ro number-of-received-octets?   uint32
```

D.2.10 o-ran-troubleshooting Module

The format for the troubleshooting module is provided below

```
module: o-ran-troubleshooting

  rpcs:
    +---x start-troubleshooting-logs
    |  +--ro output
    |     +--ro status?           enumeration
    |     +--ro failure-reason?   string
    +---x stop-troubleshooting-logs
       +--ro output
          +--ro status?           enumeration
          +--ro failure-reason?   string

  notifications:
    +---n troubleshooting-log-generated
       +--ro log-file-name*   string
```

D.2.11 o-ran-laa-operations Module

The format for the LAA operations module is provided below

```
rpcs:
    +---x start-measurements {laa:LAA}?
       +---w input
       |  +---w band-config* [band-number]
       |  |  +---w band-number       enumeration
       |  |  +---w  channel-center-frequency *   uint16
       |  +---w duration-per-channel?    uint16
       |  +---w maximum-response-time?   uint16
       +--ro output
          +--ro band-config* [band-number]
             +--ro band-number       enumeration
             +--ro carrier-center-frequency*   uint16
             +--ro status?          enumeration
             +--ro error-message?   string

  notifications:
    +---n measurement-result {laa:LAA}?
       +--ro band-result* [band-number]
          +--ro band-number              enumeration
          +--ro measurement-success?   boolean
          +--ro failure-message?       enumeration
          +--ro channel-result* [measured-channel]
             +--ro measured-channel    uint16
             +--ro occupancy-ratio?    uint8
             +--ro average-rssi?       int8
```

D.3 Interfaces Folder

D.3.1 o-ran-interfaces.yang Module

The format for the interfaces module is provided below

```
module: o-ran-interfaces
augment /if:interfaces/if:interface:
   +--rw l2-mtu?            uint16
   +--rw alias-macs*        yang:mac-address {ALIASMAC-BASED-CU-PLANE}?
   +--rw vlan-tagging?      boolean
   +--rw class-of-service
      +--rw u-plane-marking?           pcp
      +--rw c-plane-marking?           pcp
      +--rw m-plane-marking?           pcp
      +--rw s-plane-marking?           pcp
      +--rw other-marking?             pcp
      +--rw enhanced-uplane-markings* [up-marking-name]
         +--rw up-marking-name     string
         +--rw enhanced-marking?   pcp
  augment /if:interfaces/if:interface:
    +--rw base-interface?   if:interface-ref
    +--rw vlan-id?          uint16
  augment /if:interfaces/if:interface:
    +--rw mac-address?      yang:mac-address
    +--rw port-reference
    |  +--rw port-name?     -> /hw:hardware/component/name
    |  +--rw port-number?   uint8
    +--ro last-cleared?     yang:date-and-time
  augment /if:interfaces/if:interface/ip:ipv4:
    +--rw diffserv-markings {UDPIP-BASED-CU-PLANE}?
       +--rw u-plane-marking?           inet:dscp
       +--rw c-plane-marking?           inet:dscp
       +--rw s-plane-marking?           inet:dscp
       +--rw other-marking?             inet:dscp
       +--rw enhanced-uplane-markings* [up-marking-name]
          +--rw up-marking-name     string
          +--rw enhanced-marking?   inet:dscp
  augment /if:interfaces/if:interface/ip:ipv6:
    +--rw diffserv-markings {UDPIP-BASED-CU-PLANE}?
       +--rw u-plane-marking?           inet:dscp
       +--rw c-plane-marking?           inet:dscp
       +--rw s-plane-marking?           inet:dscp
       +--rw other-marking?             inet:dscp
       +--rw enhanced-uplane-markings* [up-marking-name]
          +--rw up-marking-name     string
          +--rw enhanced-marking?   inet:dscp
  augment /if:interfaces/if:interface/ip:ipv4:
    +--rw m-plane-marking?   inet:dscp
  augment /if:interfaces/if:interface/ip:ipv6:
    +--rw m-plane-marking?   inet:dscp

  rpcs:
    +---x reset-interface-counters
```

D.3.2 o-ran-processing-elements.yang Module

The format for the processing elements module is provided below

```
module: o-ran-processing-element
  +--rw processing-elements
     +--rw transport-session-type?    enumeration
     +--rw enhanced-uplane-mapping!
     |  +--rw uplane-mapping* [up-marking-name]
     |     +--rw up-marking-name        string
     |     +--rw (up-markings)?
     |        +--:(ethernet)
     |        |  +--rw up-cos-name?      -> /if:interfaces/interface/o-ran-int:class-of-service/enhanced-uplane-markings/up-marking-name
     |        +--:(ipv4)
     |        |  +--rw upv4-dscp-name?   -> /if:interfaces/interface/ip:ipv4/o-ran-int:diffserv-markings/enhanced-uplane-markings/up-marking-
name
     |        +--:(ipv6)
     |           +--rw upv6-dscp-name?   -> /if:interfaces/interface/ip:ipv6/o-ran-int:diffserv-markings/enhanced-uplane-markings/up-marking-
name
     +--rw ru-elements* [name]
        +--rw name              string
        +--rw transport-flow
           +--rw interface-name?   -> /if:interfaces/interface/name
           +--rw aliasmac-flow {o-ran-int:ALIASMAC-BASED-CU-PLANE}?
           |  +--rw ru-aliasmac-address     -> /if:interfaces/interface[if:name = current()/../../interface-name]/o-ran-int:alias-macs
           |  +--rw vlan-id?                -> /if:interfaces/interface[if:name = current()/../../interface-name]/o-ran-int:vlan-id
           |  +--rw o-du-mac-address      yang:mac-address
           +--rw eth-flow
           | +--rw ru-mac-address       -> /if:interfaces/interface[if:name = current()/../../interface-name]/o-ran-int:mac-address
           | +--rw vlan-id              -> /if:interfaces/interface[if:name = current()/../../interface-name]/o-ran-int:vlan-id
           | +--rw o-du-mac-address    yang:mac-address
           +--rw udpip-flow
             +--rw (address)
             | +--:(ru-ipv4-address)
             | | +--rw ru-ipv4-address?      -> /if:interfaces/interface[if:name = current()/../../interface-name]/ip:ipv4/address/ip
             | +--:(ru-ipv6-address)
             |    +--rw ru-ipv6-address?      -> /if:interfaces/interface[if:name = current()/../../interface-name]/ip:ipv6/address/ip
             +--rw o-du-ip-address           inet:ip-address
             +--rw ru-ephemeral-udp-port       inet:port-number
             +--rw o-du-ephemeral-udp-port    inet:port-number
             +--rw ecpri-destination-udp       inet:port-number
```

D.3.3 o-ran-transceiver.yang Module

The format for the (SFP) transciver module is provided below

```
module: o-ran-transceiver
  +--rw port-transceivers
     +--rw port-transceiver-data* [interface-name port-number]
        +--rw interface-name            -> /if:interfaces/interface/name
        +--rw port-number               -> /if:interfaces/interface[if:name = current()/../interface-name]/o-ran-int:port-reference/port-number
        +--rw name?                    string
        +--ro present                  boolean
        +--ro vendor-id?               string
        +--ro vendor-part?             string
        +--ro vendor-rev?              string
        +--ro serial-no?               string
        +--ro SFF8472-compliance-code?   enumeration
        +--ro connector-type?           enumeration
        +--ro nominal-bitrate?          uint32
        +--ro low-bitrate-margin?       uint8
        +--ro high-bitrate-margin?      uint8
        +--ro rx-power-type?            enumeration
        +--ro rx-power?                decimal64
        +--ro tx-power?                decimal64
        +--ro tx-bias-current?          decimal64
        +--ro voltage?                 decimal64
        +--ro temperature?             decimal64
```

D.3.4 -ran-mplane-int.yang Module

The format for the management plane interface module is provided below

```
module: o-ran-mplane-int
  +--rw mplane-info
     +--rw searchable-mplane-access-vlans-info
     | +--rw searchable-access-vlans*   vlan-id
     | +--rw vlan-range
     |    +--rw lowest-vlan-id?    vlan-id
     |    +--rw highest-vlan-id?   vlan-id
     +--rw m-plane-interfaces
        +--rw m-plane-sub-interfaces* [interface-name sub-interface]
        | +--rw interface-name    -> /if:interfaces/interface/name
        | +--rw sub-interface     -> /if:interfaces/interface[if:name = current()/../interface-name]/o-ran-int:vlan-id
        | +--rw client-info
        |    +--rw mplane-ipv4-info* [mplane-ipv4]
        |    | +--rw mplane-ipv4    inet:ipv4-address
        |    | +--rw port?         inet:port-number
        |    +--rw mplane-ipv6-info* [mplane-ipv6]
        |    | +--rw mplane-ipv6    inet:ipv6-address
        |    | +--rw port?         inet:port-number
        |    +--rw mplane-fqdn*       inet:domain-name
        +--rw m-plane-ssh-ports
           +--rw call-home-ssh-port?   inet:port-number
           +--rw server-ssh-port?      inet:port-number
```

D.3.5 o-ran-dhcp.yang Module

The format for the dhcp module is provided below.

```
module: o-ran-dhcp
  +--ro dhcp
```

```
    +--ro interfaces* [interface]
    |  +--ro interface    if:interface-ref
    |  +--ro dhcpv4
    |  |  +--ro client-id?                string
    |  |  +--ro dhcp-server-identifier?   inet:ip-address
    |  |  +--ro domain-name?              string
    |  |  +--ro domain-name-servers*      inet:ip-address
    |  |  +--ro interface-mtu?            uint32
    |  |  +--ro default-gateways*         inet:ip-address
    |  |  +--ro netconf-clients* [client]
    |  |     +--ro client           netconf-client-id
    |  |     +--ro optional-port?   inet:port-number
    |  +--ro dhcpv6
    |     +--ro dhcp-client-identifier
    |     |  +--ro type-code?                         uint16
    |     |  +--ro (duid-type)?
    |     |     +--:(duid-llt)
    |     |     |  +--ro duid-llt-hardware-type?      uint16
    |     |     |  +--ro duid-llt-time?               yang:timeticks
    |     |     |  +--ro duid-llt-link-layer-addr?    yang:mac-address
    |     |     +--:(duid-en)
    |     |     |  +--ro duid-en-enterprise-number?   uint32
    |     |     |  +--ro duid-en-identifier?          string
    |     |     +--:(duid-ll)
    |     |     |  +--ro duid-ll-hardware-type?       uint16
    |     |     |  +--ro duid-ll-link-layer-addr?     yang:mac-address
    |     |     +--:(duid-uuid)
    |     |     |  +--ro uuid?                        yang:uuid
    |     |     +--:(duid-unknown)
    |     |        +--ro data?                        binary
    |     +--ro dhcp-server-identifier
    |     |  +--ro type-code?                         uint16
    |     |  +--ro (duid-type)?
    |     |     +--:(duid-llt)
    |     |     |  +--ro duid-llt-hardware-type?      uint16
    |     |     |  +--ro duid-llt-time?               yang:timeticks
    |     |     |  +--ro duid-llt-link-layer-addr?    yang:mac-address
    |     |     +--:(duid-en)
    |     |     |  +--ro duid-en-enterprise-number?   uint32
    |     |     |  +--ro duid-en-identifier?          string
    |     |     +--:(duid-ll)
    |     |     |  +--ro duid-ll-hardware-type?       uint16
    |     |     |  +--ro duid-ll-link-layer-addr?     yang:mac-address
    |     |     +--:(duid-uuid)
    |     |     |  +--ro uuid?                        yang:uuid
    |     |     +--:(duid-unknown)
    |     |        +--ro data?                        binary
    |     +--ro domain-name?              string
    |     +--ro domain-name-servers*      inet:ip-address
    |     +--ro netconf-clients* [client]
    |        +--ro client           netconf-client-id
    |        +--ro optional-port?   inet:port-number
    +--ro m-plane-dhcp
       +--ro private-enterprise-number?   uint16
       +--ro vendor-class-data?           string
```

D.3.6 o-ran-externalio.yang Module

The format for the external input/output module is provided below

```
module: o-ran-externalio
  +--rw external-io
     +--ro input* [name]
     |  +--ro name       string
     |  +--ro port-in?   uint8
     |  +--ro line-in?   boolean
     +--ro output* [name]
     |  +--ro name        string
     |  +--ro port-out    uint8
     +--rw output-setting* [name]
        +--rw name        -> /external-io/output/name
        +--rw line-out?   boolean
  notifications:
    +---n external-input-change
       +--ro current-input-notification
          +--ro external-input* [name]
        +--ro name       -> /external-io/input/name
        +--ro io-port?   -> /external-io/input/port-in
        +--ro line-in?   -> /external-io/input/line-in
```

D.3.7 o-ran-ald-port.yang Module

The format for the Antenna Line Device module is provided below

```
module: o-ran-ald-port
  +--rw ald-ports-io
     +--ro over-current-supported?   boolean
     +--ro ald-port* [name]
     | +--ro name                    string
     | +--ro port-id                 uint8
     | +--ro dc-control-support     boolean
     | +--ro dc-enabled-status?     boolean
     | +--ro supported-connector    enumeration
     +--rw ald-port-dc-control* [name]
        +--rw name          -> /ald-ports-io/ald-port/name
        +--rw dc-enabled?   boolean

  notifications:
    +---n overcurrent-report {OVERCURRENT-SUPPORTED}?
       +--ro overload-condition
          +--ro overloaded-ports*   -> /ald-ports-io/ald-port/name
```

D.4 Sync Folder

D.4.1 o-ran-sync.yang Module

The format for the synchronization module is provided below

```
module: o-ran-sync
  +--rw sync
     +--ro sync-status
     | +--ro sync-state                    enumeration
     | +--ro supported-reference-types* [item]
     |    +--ro item    enumeration
     +--ro sync-capability
     | +--ro sync-t-tsc    enumeration
     +--rw ptp-config
     | +--rw domain-number?            uint8
     | +--rw accepted-clock-classes* [clock-classes]
     | | +--rw clock-classes    uint8
     | +--rw ptp-profile?              enumeration
     | +--rw g-8275-1-config
     | | +--rw multicast-mac-address?   enumeration
     | | +--rw delay-asymmetry?        int16
     | +--rw g-8275-2-config
     |    +--rw local-ip-port?              -> /if:interfaces/interface/name
     |    +--rw master-ip-configuration* [local-priority]
     |    | +--rw local-priority    uint8
     |    | +--rw ip-address?       string
     |    +--rw log-inter-sync-period?       int8
     |    +--rw log-inter-announce-period?   int8
     +--rw ptp-status
     | +--rw reporting-period?           uint8
     | +--ro lock-state?                 enumeration
     | +--ro clock-class?                uint8
     | +--ro clock-identity?             string
     | +--ro partial-timing-supported?   boolean
     | +--ro sources* [local-port-number]
     |    +--ro local-port-number              -> /if:interfaces/interface/o-ran-int:port-reference/port-number
     |    +--ro state?                       enumeration
     |    +--ro two-step-flag?                 boolean
     |    +--ro leap61?                      boolean
     |    +--ro leap59?                      boolean
     |    +--ro current-utc-offset-valid?     boolean
     |    +--ro ptp-timescale?                boolean
     |    +--ro time-traceable?               boolean
     |    +--ro frequency-traceable?          boolean
     |    +--ro source-clock-identity?        string
     |    +--ro source-port-number?           uint16
     |    +--ro current-utc-offset?          int16
```

```
    |     +--ro priority1?                  uint8
    |     +--ro clock-class?                uint8
    |     +--ro clock-accuracy?             uint8
    |     +--ro offset-scaled-log-variance?   uint16
    |     +--ro priority2?                  uint8
    |     +--ro grandmaster-clock-identity?   string
    |     +--ro steps-removed?              uint16
    |     +--ro time-source?                uint8
    +--rw synce-config
    | +--rw acceptance-list-of-ssm*   enumeration
    | +--rw ssm-timeout?             uint16
    +--rw synce-status
    | +--rw reporting-period?   uint8
    | +--ro lock-state?         enumeration
    | +--ro sources* [local-port-number]
    |     +--ro local-port-number    -> /if:interfaces/interface/o-ran-int:port-reference/port-number
    |     +--ro state?               enumeration
    |     +--ro quality-level?       uint8
    +--rw gnss-config {GNSS}?
    | +--rw enable?                      boolean
    | +--rw satellite-constelation-list*    enumeration
    | +--rw polarity?                    enumeration
    | +--rw cable-delay?                 uint16
    | +--rw anti-jam-enable?             boolean {ANTI-JAM}?
    +--rw gnss-status {GNSS}?
      +--rw reporting-period?   uint8
      +--ro name?               string
      +--ro gnss-sync-status?   enumeration
      +--ro gnss-data
        +--ro satellites-tracked?   uint8
        +--ro location
          +--ro altitude?    int64
          +--ro latitude?    geographic-coordinate-degree
          +--ro longitude?   geographic-coordinate-degree

 notifications:
  +---n synchronization-state-change
  | +--ro sync-state?   -> /sync/sync-status/sync-state
  +---n ptp-state-change
  | +--ro ptp-state?   -> /sync/ptp-status/lock-state
  +---n synce-state-change
  | +--ro synce-state?   -> /sync/synce-status/lock-state
  +---n gnss-state-change {GNSS}?
    +--ro gnss-state?   -> /sync/gnss-status/gnss-sync-status
```

D.5 Radio Folder

D.5.1 o-ran-module-cap.yang Module

The format for the module capabilitites module is provided below

```


module: o-ran-module-cap
  +--ro module-capability
    +--ro ru-capabilities
    | +--ro ru-supported-category?                  enumeration
    | +--ro number-of-ru-ports?                     uint8
    | +--ro number-of-spatial-streams?              uint8
    | +--ro max-power-per-pa-antenna?               decimal64
    | +--ro min-power-per-pa-antenna?               decimal64
    | +--ro fronthaul-split-option?                 uint8
    | +--ro format-of-iq-sample
    | | +--ro dynamic-compression-supported?             boolean
    | | +--ro realtime-variable-bit-width-supported?     boolean
    | | +--ro compression-method-supported* []
    | | +--ro variable-bit-width-per-channel-supported?   boolean
    | | +--ro syminc-supported?                         boolean
    | +--ro ul-mixed-num-required-guard-rbs* [scs-a scs-b]
    | | +--ro scs-a                  enumeration
    | | +--ro scs-b                  enumeration
    | | +--ro number-of-guard-rbs-ul?   uint8
    | +--ro dl-mixed-num-required-guard-rbs* [scs-a scs-b]
    | | +--ro scs-a                  enumeration
    | |  +--ro scs-b                    enumeration
    | |  +--ro number-of-guard-rbs-dl?  uint8
    |  +--ro energy-saving-by-transmission-blanks           boolean
    |  +--ro dynamic-transport-delay-management-supported   boolean
    +--ro band-capabilities* [band-number]
       +--ro band-number                 uint16
       +--ro sub-band-info {o-ran-module-cap:LAA}?
       |  +--ro sub-band-frequency-ranges* [sub-band]
       |  |  +--ro sub-band                    sub-band-string
       |  |  +--ro max-supported-frequency-dl?   uint64
       |  |  +--ro min-supported-frequency-dl?   uint64
       |  +--ro number-of-laa-scells?        uint8
       |  +--ro maximum-laa-buffer-size?     uint16
       |  +--ro maximum-processing-time?     uint16
       |  +--ro self-configure?              boolean
       +--ro max-supported-frequency-dl?   uint64
       +--ro min-supported-frequency-dl?   uint64
       +--ro max-supported-bandwidth-dl?   uint64
       +--ro max-num-carriers-dl?          uint32
       +--ro max-carrier-bandwidth-dl?     uint64
       +--ro min-carrier-bandwidth-dl?     uint64
       +--ro max-supported-frequency-ul?   uint64
       +--ro min-supported-frequency-ul?   uint64
       +--ro max-supported-bandwidth-ul?   uint64
       +--ro max-num-carriers-ul?          uint32
       +--ro max-carrier-bandwidth-ul?     uint64
       +--ro min-carrier-bandwidth-ul?     uint64
       +--ro max-num-component-carriers?   uint8
       +--ro max-num-bands?                uint16
       +--ro max-num-sectors?              uint8
       +--ro max-power-per-antenna?        decimal64
       +--ro min-power-per-antenna?        decimal64
       +--ro codebook-configuration_ng?    uint8
       +--ro codebook-configuration_n1?    uint8
       +--ro codebook-configuration_n2?    uint8
```

D.5.2 o-ran-delay-management.yang Module

The format for the delay management module is provided below

```
module: o-ran-delay-management
  +--rw delay-management
    +--rw bandwidth-scs-delay-state* [bandwidth subcarrier-spacing]
    |  +--rw bandwidth             uint32
    |  +--rw subcarrier-spacing    uint32
    |  +--ro ru-delay-profile
    |     +--ro t2a-min-up       uint32
    |     +--ro t2a-max-up       uint32
    |     +--ro t2a-min-cp-dl    uint32
    |     +--ro t2a-max-cp-dl    uint32
    |     +--ro tcp-adv-dl       uint32
    |     +--ro ta3-min          uint32
    |     +--ro ta3-max          uint32
    |     +--ro t2a-min-cp-ul    uint32
    |     +--ro t2a-max-cp-ul    uint32
    +--rw adaptive-delay-configuration {ADAPTIVE-RU-PROFILE}?
       +--rw bandwidth-scs-delay-state* [bandwidth subcarrier-spacing]
       |  +--rw bandwidth             uint32
       |  +--rw subcarrier-spacing    uint32
       |  +--rw O-DU-delay-profile
       |     +--rw t1a-max-up?   uint32
       |     +--rw tx-max?       uint32
       |     +--rw ta4-max?      uint32
       |     +--rw rx-max?       uint32
       +--rw transport-delay
          +--rw t12-min?   uint32
          +--rw t34-min?   uint32
          +--rw t12-max?   uint32
          +--rw t34-max?   uint32
```

D.5.3 o-ran-beamforming.yang Module

The format for the beamforming module is provided below

```
module: o-ran-beamforming
```

```
+--ro beamforming-config
   +--ro per-band-config* [band-number]
   |  +--ro band-number         -> /mcap:module-capability/band-capabilities/band-number
   |  +--ro tx-array*           -> /up:user-plane-configuration/tx-arrays/name
   |  +--ro rx-array*           -> /up:user-plane-configuration/rx-arrays/name
   |  +--ro static-properties
   |  |  +--ro rt-bf-weights-update-support?   boolean
   |  |  +--ro rt-bf-weights-update-support?   boolean
   |  |  +--ro (beamforming-type)?
   |  |  |  +--:(frequency)
   |  |  |  |  +--ro frequency-domain-beams
   |  |  |  |     +--ro max-number-of-beam-ids                  uint16
   |  |  |  |     +--ro initial-beam-id                         uint16
   |  |  |  |     +--ro compression-type                        enumeration
   |  |  |  |     +--ro bitwidth?                               uint8
   |  |  |  |     +--ro (compression-format)?
   |  |  |  |        +--:(no-compresison)
   |  |  |  |        +--:(block-floating-point)
   |  |  |  |        |  +--ro exponent?                         uint8
   |  |  |  |        +--:(block-scaling)
   |  |  |  |        |  +--ro block-scalar?                     uint8
   |  |  |  |        +--:(u-law)
   |  |  |  |        |  +--ro comp-bit-width?                   uint8
   |  |  |  |        |  +--ro comp-shift?                       uint8
   |  |  |  |        +--:(beam-space-compression)
   |  |  |  |        |  +--ro active-beam-space-coeficient-mask*   uint8
   |  |  |  |        |  +--ro block-scaler?                     uint8
   |  |  |  |        +--:(modulation-compression)
   |  |  |  |           +--ro scf?                              uint8
   |  |  |  |           +--ro mod-comp-scaler?                  uint16
   |  |  |  +--:(time)
   |  |  |  |  +--ro time-domain-beams
   |  |  |  |     +--ro max-number-of-beam-ids                  uint16
   |  |  |  |     +--ro initial-beam-id                         uint16
   |  |  |  |     +--ro frequency-granularity                   enumeration
   |  |  |  |     +--ro time-granularity                        enumeration
   |  |  |  |     +--ro compression-type                        enumeration
   |  |  |  |     +--ro bitwidth?                               uint8
   |  |  |  |     +--ro (compression-format)?
   |  |  |  |        +--:(no-compresison)
   |  |  |  |        +--:(block-floating-point)
   |  |  |  |        |  +--ro exponent?                         uint8
   |  |  |  |        +--:(block-scaling)
   |  |  |  |        |  +--ro block-scalar?                     uint8
   |  |  |  |        +--:(u-law)
   |  |  |  |        |  +--ro comp-bit-width?                   uint8
   |  |  |  |        |  +--ro comp-shift?                       uint8
   |  |  |  |        +--:(beam-space-compression)
   |  |  |  |        |  +--ro active-beam-space-coeficient-mask*   uint8
   |  |  |  |        |  +--ro block-scaler?                     uint8
   |  |  |  |        +--:(modulation-compression)
   |  |  |  |           +--ro scf?                              uint8
   |  |  |  |           +--ro mod-comp-scaler?                  uint16
   |  |  |  +--:(hybrid)
   |  |  |     +--ro hybrid-beams
   |  |  |        +--ro max-number-of-beam-ids                  uint16
   |  |  |        +--ro initial-beam-id                         uint16
   |  |  |        +--ro frequency-granularity                   enumeration
   |  |  |        +--ro time-granularity                        enumeration
   |  |  |        +--ro compression-type                        enumeration
   |  |  |        +--ro bitwidth?                               uint8
   |  |  |        +--ro (compression-format)?
   |  |  |           +--:(no-compresison)
   |  |  |           +--:(block-floating-point)
   |  |  |           |  +--ro exponent?                         uint8
   |  |  |           +--:(block-scaling)
   |  |  |           |  +--ro block-scalar?                     uint8
   |  |  |           +--:(u-law)
   |  |  |           |  +--ro comp-bit-width?                   uint8
   |  |  |           |  +--ro comp-shift?                       uint8
   |  |  |           +--:(beam-space-compression)
   |  |  |           |  +--ro active-beam-space-coeficient-mask*   uint8
   |  |  |           |  +--ro block-scaler?                     uint8
   |  |  |           +--:(modulation-compression)
   |  |  |              +--ro scf?                              uint8
   |  |  |              +--ro mod-comp-scaler?                  uint16
   |  |  +--ro number-of-beams?                  uint16
```

```
  | +--ro beam-information
  |    +--ro number-of-beamforming-properties?   uint16
  |    +--ro beamforming-properties* [beam-id]
  |       +--ro beam-id                    uint16
  |       +--ro beamforming-property
  |          +--ro beam-type?                  enumeration
  |          +--ro beam-group-id?               uint16
  |          +--ro coarse-fine-beam-relation*   beam-reference
  |          +--ro neighbour-beams*             beam-reference
  +--ro ue-specific-beamforming!
  | +--ro max-number-of-ues?   uint8
  +--ro operational-properties {MODIFY-BF-CONFIG}?
  | +--ro number-of-writeable-beamforming-files    uint8
  | +--ro update-bf-non-delete?                    boolean
  | +--ro persistent-bf-files?                     boolean
  +--ro beamforming-through-attributes-supported?        boolean
  +--ro beamforming-through-ue-channel-info-supported?   boolean

rpcs:
  +---x activate-beamforming-config {MODIFY-BF-CONFIG}?
     +---w input
     | +---w beamforming-config-file    string
     | +---w band-number?               -> /mcap:module-capability/band-capabilities/band-number
     +--ro output
        +--ro status           enumeration
        +--ro error-message?   string

notifications:
  +---n beamforming-information-update
     +--ro band-number?   -> /mcap:module-capability/band-capabilities/band-number
```

D.5.4 o-ran-laa.yang Module

The format for the LAA module is provided below

```
+--rw laa-config
   +--rw number-of-laa-scells?                  uint8
   +--rw multi-carrier-type?                    enumeration
   +--rw multi-carrier-tx?                      boolean
   +--rw multi-carrier-freeze?                  boolean
   +--rw laa-ending-dwpts-supported?            boolean
   +--rw laa-starting-in-second-slot-supported?   boolean
```

Annex ZZZ O-RAN Adopter License Agreement

BY DOWNLOADING, USING OR OTHERWISE ACCESSING ANY O-RAN SPECIFICATION, ADOPTER AGREES TO THE TERMS OF THIS AGREEMENT.

This O-RAN Adopter License Agreement (the “Agreement”) is made by and between the O-RAN Alliance and the entity that downloads, uses or otherwise accesses any O-RAN Specification, including its Affiliates (the “Adopter”).

This is a license agreement for entities who wish to adopt any O-RAN Specification.

Section 1: DEFINITIONS

1.1 “Affiliate” means an entity that directly or indirectly controls, is controlled by, or is under common control with another entity, so long as such control exists. For the purpose of this Section, “Control” means beneficial ownership of fifty (50%) percent or more of the voting stock or equity in an entity.

1.2 “Compliant Implementation” means any system, device, method or operation (whether implemented in hardware, software or combinations thereof) that fully conforms to a Final Specification.

1.3 “Adopter(s)” means all entities, who are not Members, Contributors or Academic Contributors, including their Affiliates, who wish to download, use or otherwise access O-RAN Specifications.

1.4 “Minor Update” means an update or revision to an O-RAN Specification published by O-RAN Alliance that does not add any significant new features or functionality and remains interoperable with the prior version of an O-RAN Specification. The term “O-RAN Specifications” includes Minor Updates.

1.5 “Necessary Claims” means those claims of all present and future patents and patent applications, other than design patents and design registrations, throughout the world, which (i) are owned or otherwise licensable by a Member, Contributor or Academic Contributor during the term of its Member, Contributor or Academic Contributorship; (ii) such Member, Contributor or Academic Contributor has the right to grant a license without the payment of consideration to a third party; and (iii) are necessarily infringed by a Compliant Implementation (without considering any Contributions not included in the Final Specification). A claim is necessarily infringed only when it is not possible on technical (but not commercial) grounds, taking into account normal technical practice and the state of the art generally available at the date any Final Specification was published by the O-RAN Alliance or the date the patent claim first came into existence, whichever last occurred, to make, sell, lease, otherwise dispose of, repair, use or operate a Compliant Implementation without infringing that claim. For the avoidance of doubt in exceptional cases where a Final Specification can only be implemented by technical solutions, all of which infringe patent claims, all such patent claims shall be considered Necessary Claims.

1.6 “Defensive Suspension” means for the purposes of any license grant pursuant to Section 3, Member, Contributor, Academic Contributor, Adopter, or any of their Affiliates, may have the discretion to include in their license a term allowing the licensor to suspend the license against a licensee who brings a patent infringement suit against the licensing Member, Contributor, Academic Contributor, Adopter, or any of their Affiliates.

Section 2: COPYRIGHT LICENSE

2.1 Subject to the terms and conditions of this Agreement, O-RAN Alliance hereby grants to Adopter a nonexclusive, nontransferable, irrevocable, non-sublicensable, worldwide copyright license to obtain, use and modify O-RAN Specifications, but not to further distribute such O-RAN Specification in any modified or unmodified way, solely in furtherance of implementations of an ORAN Specification.

2.2 Adopter shall not use O-RAN Specifications except as expressly set forth in this Agreement or in a separate written agreement with O-RAN Alliance.

Section 3: FRAND LICENSE

3.1 Members, Contributors and Academic Contributors and their Affiliates are prepared to grant based on a separate Patent License Agreement to each Adopter under Fair Reasonable And Non- Discriminatory (FRAND) terms and conditions with or without compensation (royalties) a nonexclusive, non-transferable, irrevocable (but subject to Defensive Suspension), non-sublicensable, worldwide patent license under their Necessary Claims to make, have made, use, import, offer to sell, lease, sell and otherwise distribute Compliant Implementations; provided, however, that such license shall not extend: (a) to any part or function of a product in which a Compliant Implementation is incorporated that is not itself part of the Compliant Implementation; or (b) to any Adopter if that Adopter is not making a reciprocal grant to Members, Contributors and Academic Contributors, as set forth in Section 3.3. For the avoidance of doubt, the foregoing licensing commitment includes the distribution by the Adopter’s distributors and the use by the Adopter’s customers of such licensed Compliant Implementations.

3.2 Notwithstanding the above, if any Member, Contributor or Academic Contributor, Adopter or their Affiliates has reserved the right to charge a FRAND royalty or other fee for its license of Necessary Claims to Adopter, then Adopter is entitled to charge a FRAND royalty or other fee to such Member, Contributor or Academic Contributor, Adopter and its Affiliates for its license of Necessary Claims to its licensees.

3.3 Adopter, on behalf of itself and its Affiliates, shall be prepared to grant based on a separate Patent License Agreement to each Members, Contributors, Academic Contributors, Adopters and their Affiliates under Fair Reasonable And Non-Discriminatory (FRAND) terms and conditions with or without compensation (royalties) a nonexclusive, non-transferable, irrevocable (but subject to Defensive Suspension), non-sublicensable, worldwide patent license under their Necessary Claims to make, have made, use, import, offer to sell, lease, sell and otherwise distribute Compliant Implementations; provided, however, that such license will not extend: (a) to any part or function of a product in which a Compliant Implementation is incorporated that is not itself part of the Compliant Implementation; or (b) to any Members, Contributors, Academic Contributors, Adopters and their Affiliates that is not making a reciprocal grant to Adopter, as set forth in Section 3.1. For the avoidance of doubt, the foregoing licensing commitment includes the distribution by the Members’, Contributors’, Academic Contributors’, Adopters’ and their Affiliates’ distributors and the use by the Members’, Contributors’, Academic Contributors’, Adopters’ and their Affiliates’ customers of such licensed Compliant Implementations.

Section 4: TERM AND TERMINATION

4.1 This Agreement shall remain in force, unless early terminated according to this Section 4.

4.2 O-RAN Alliance on behalf of its Members, Contributors and Academic Contributors may terminate this Agreement if Adopter materially breaches this Agreement and does not cure or is not capable of curing such breach within thirty (30) days after being given notice specifying the breach.

4.3 Sections 1, 3, 5 - 11 of this Agreement shall survive any termination of this Agreement. Under surviving Section 3, after termination of this Agreement, Adopter will continue to grant licenses (a) to entities who become Adopters after the date of termination; and (b) for future versions of ORAN Specifications that are backwards compatible with the version that was current as of the date of termination.

Section 5: CONFIDENTIALITY

Adopter will use the same care and discretion to avoid disclosure, publication, and dissemination of O-RAN Specifications to third parties, as Adopter employs with its own confidential information, but no less than reasonable care. Any disclosure by Adopter to its Affiliates, contractors and consultants should be subject to an obligation of confidentiality at least as restrictive as those contained in this Section. The foregoing obligation shall not apply to any information which is: (1) rightfully known by Adopter without any limitation on use or disclosure prior to disclosure; (2) publicly available through no fault of Adopter; (3) rightfully received without a duty of confidentiality; (4) disclosed by O-RAN Alliance or a Member, Contributor or Academic Contributor to a third party without a duty of confidentiality on such third party; (5) independently developed by Adopter; (6) disclosed pursuant to the order of a court or other authorized governmental body, or as required by law, provided that Adopter provides reasonable prior written notice to O-RAN Alliance, and cooperates with O-RAN Alliance and/or the applicable Member, Contributor or Academic Contributor to have the opportunity to oppose any such order; or (7) disclosed by Adopter with O-RAN Alliance's prior written approval.

Section 6: INDEMNIFICATION

Adopter shall indemnify, defend, and hold harmless the O-RAN Alliance, its Members, Contributors or Academic Contributors, and their employees, and agents and their respective successors, heirs and assigns (the "Indemnitees"), against any liability, damage, loss, or expense (including reasonable attorneys' fees and expenses) incurred by or imposed upon any of the Indemnitees in connection with any claims, suits, investigations, actions, demands or judgments arising out of Adopter's use of the licensed O-RAN Specifications or Adopter's commercialization of products that comply with O-RAN Specifications.

Section 7: LIMITATIONS ON LIABILITY; NO WARRANTY

EXCEPT FOR BREACH OF CONFIDENTIALITY, ADOPTER'S BREACH OF SECTION 3, AND ADOPTER'S INDEMNIFICATION OBLIGATIONS, IN NO EVENT SHALL ANY PARTY BE LIABLE TO ANY OTHER PARTY OR THIRD PARTY FOR ANY INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES RESULTING FROM ITS PERFORMANCE OR NON-PERFORMANCE UNDER THIS AGREEMENT, IN EACH CASE WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, AND WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

O-RAN SPECIFICATIONS ARE PROVIDED "AS IS" WITH NO WARRANTIES OR CONDITIONS WHATSOEVER, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. THE O-RAN ALLIANCE AND THE MEMBERS, CONTRIBUTORS OR ACADEMIC CONTRIBUTORS EXPRESSLY DISCLAIM ANY WARRANTY OR CONDITION OF MERCHANTABILITY, SECURITY, SATISFACTORY QUALITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, ERROR-FREE OPERATION, OR ANY WARRANTY OR CONDITION FOR O-RAN SPECIFICATIONS.

Section 8: ASSIGNMENT

Adopter may not assign the Agreement or any of its rights or obligations under this Agreement or make any grants or other sublicenses to this Agreement, except as expressly authorized hereunder, without having first received the prior, written consent of the O-RAN Alliance, which consent may be withheld in O-RAN Alliance's sole discretion. O-RAN Alliance may freely assign this Agreement.

Section 9: THIRD-PARTY BENEFICIARY RIGHTS

Adopter acknowledges and agrees that Members, Contributors and Academic Contributors (including future Members, Contributors and Academic Contributors) are entitled to rights as a third-party beneficiary under this Agreement, including as licensees under Section 3.

Section 10: BINDING ON AFFILIATES

Execution of this Agreement by Adopter in its capacity as a legal entity or association constitutes that legal entity's or association's agreement that its Affiliates are likewise bound to the obligations that are applicable to Adopter hereunder and are also entitled to the benefits of the rights of Adopter hereunder.

Section 11: GENERAL

This Agreement is governed by the laws of Germany without regard to its conflict or choice of law provisions.

This Agreement constitutes the entire agreement between the parties as to its express subject matter and expressly supersedes and replaces any prior or contemporaneous agreements between the parties, whether written or oral, relating to the subject matter of this Agreement.

Adopter, on behalf of itself and its Affiliates, agrees to comply at all times with all applicable laws, rules and regulations with respect to its and its Affiliates' performance under this Agreement, including without limitation, export control and antitrust laws. Without limiting the generality of the foregoing, Adopter acknowledges that this Agreement prohibits any communication that would violate the antitrust laws.

By execution hereof, no form of any partnership, joint venture or other special relationship is created between Adopter, or O-RAN Alliance or its Members, Contributors or Academic Contributors. Except as expressly set forth in this Agreement, no party is authorized to make any commitment on behalf of Adopter, or O-RAN Alliance or its Members, Contributors or Academic Contributors.

In the event that any provision of this Agreement conflicts with governing law or if any provision is held to be null, void or otherwise ineffective or invalid by a court of competent jurisdiction, (i) such provisions will be deemed stricken from the contract, and (ii) the remaining terms, provisions, covenants and restrictions of this Agreement will remain in full force and effect.

Any failure by a party or third party beneficiary to insist upon or enforce performance by another party of any of the provisions of this Agreement or to exercise any rights or remedies under this Agreement or otherwise by law shall not be construed as a waiver or relinquishment to any extent of the other parties' or third party beneficiary's right to assert or rely upon any such provision, right or remedy in that or any other instance; rather the same shall be and remain in full force and effect.