ERICSSON

COM NETCONF Interface Base

Common Operation and Maintenance

INTERWORK DESCRIPTION

2/155 19-CAA 901 2587/6 Uen A1

Contents

1 Introduction

Note:

INSTRUCTION FOR DOCUMENT REUSE: This is a BASE document that describes the NETCONF functions provided by COM. This document is not to be reused in the COM user products product information without modification. COM user products must create a product-specific variant of this document according to the instructions. Provide information on the following product-specific properties:

- Indicate limitations and remove description on not supported COM functions.
- Replace COM-specific document references with product-specific ones.
- Update the document type and title to comply with the relevant rules.
- Remove the document reuse instructions from the document. These instructions are all in the form of **Notes** including the text, **INSTRUCTION FOR DOCUMENT REUSE:**.

This document describes the functions provided in the Common Operation and Maintenance (COM) Network Configuration (NETCONF) interface.

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Replace with the final NETCONF interface name.

1.1 Prerequisites

This section describes the prerequisites for using the NETCONF interface.

1.1.1 Hardware and Software Required

The following software is required:

- A standard Secure Shell (SSH) version 2 client.
- TLS 1.2 support if the NETCONF over TLS function is intended to be used.

1.1.2 Conditions

Before using the NETCONF interface, the following conditions must apply:

- Access to the Operation and Maintenance (O&M) IP address of the Managed Element.
- Access to the NETCONF port is enabled in the firewalls.
- A valid O&M user account.
- The user has prior knowledge of generic O&M concepts. For more information, refer to the following documents:
  - *Glossary of Terms and Acronyms*
  - *COM Management Guide*

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Replace with product-specific documents.

2 Overview

The NETCONF protocol provides mechanisms to modify the configuration of network devices, and to monitor status and statistics. It uses an XML-based data encoding for the configuration data and the protocol messages. The NETCONF protocol operations are realized as Remote Procedure Calls (RPCs).

The key properties of the NETCONF interface are as follows:

Access control

Access control: operations are subject of authentication and authorization, refer to *COM Management Guide*.

Note: INSTRUCTION FOR DOCUMENT REUSE: Replace with product-specific documents.

Data model

Configuration and state data is represented as Managed Object (MO) instances, attributes, and actions in COM NETCONF-specific XML format in the COM NETCONF messages.

For information on COM NETCONF XML, see Section 6 on page 21.

Datastore

The running and startup data store are supported based on the operation. See the operations in Section 7 on page 31.

Log

All operations are logged and the log records contain information on username, session ID, operation time, name with parameters, and operation responses. For details, see the product-specific Audit Log documentation.

Note: INSTRUCTION FOR DOCUMENT REUSE: Replace with product-specific documents.

Model-driven

The MO class, attribute type, and action properties are defined by the information models that are described in the Managed Object Model (MOM) documentation.

Note: INSTRUCTION FOR DOCUMENT REUSE: Replace with product-specific documents.

As a limitation, COM NETCONF does not use capabilities to announce the set of data models that the Managed Element implements.

Multiple sessions

Multiple parallel sessions are supported.

Namespaces The NETCONF responses generated from COM contain an XML namespace, defined by XML attribute `xmlns`. The namespace is defined in each MOM. However, the `xmlns` is ignored by COM in requests sent from the Management System. The MO Distinguished Name (DN) is in itself enough to identify the MO instance and its class.

Security User authentication and secure NETCONF connection and transport over SSL is provided by the SSH daemon or the TLS proxy component.

Standard compliance

Compliant to NETCONF standards, see Section 3 on page 5.

Transactions Configuration changes are applied transactionally. Thus, it is ensured that all or none of the operations are executed. For details, see Section 11 on page 103.

UTF-8 The complete UTF-8 character range is supported.

Note: The examples in this document are based on models that are subject of change.

3 Standard Compliance

The COM NETCONF interface is standard compliant according to Table 1.

Table 1 Standard Compliance

RFC	Compliance
NETCONF protocol RFCs: • RFC 4741 – NETCONF Configuration Protocol(1) • RFC 6241 – Network Configuration Protocol (NETCONF)	Partially compliant. For details, see Table 2 or Section 15.1 Compliance to RFC 4741 and RFC 6241 on page 125.
NETCONF over SSH mapping RFCs (these are mandatory to implement): • RFC 4742 – Using the NETCONF Configuration Protocol over Secure SHell (SSH)(2) • RFC 6242 – Using the NETCONF Protocol over Secure Shell (SSH)	Compliant to RFC 4742. Partly compliant to RFC 6242.
NETCONF notification RFCs: • RFC 5277 – NETCONF Event Notifications • RFC 5539 – NETCONF over Transport Layer Security (TLS)	Partly compliant.

(1) RFC 4741 is obsoleted by RFC 6241.
(2) RFC 4742 is obsoleted by RFC 6242.

A summary of the NETCONF capabilities is shown in Table 2.

Table 2 NETCONF Capabilities

RFC	Capability Identifier	Compliance
4741	`urn:ietf:params:netconf:base:1.0`	Partly compliant. Supported operations: • `<get>` • `<get-config>` • `<close-session>` • `<kill-session>` • `<copy-config>` • `<delete-config>` Supported with limitation: • `<edit-config>`(1) Not supported operations: • `<lock>`(2) • `<unlock>`
RFC 6241	`urn:ietf:params:netconf:base:1.1`	Not compliant.
4741, 6241	`urn:ietf:params:netconf:capability:writable-running:1.0`	Compliant.

Table 2 NETCONF Capabilities

RFC	Capability Identifier	Compliance
4741, 6241	urn:ietf:params:netconf:capability:**candidate**:1.0	Not compliant.
4741, 6241	urn:ietf:params:netconf:capability:**confirmed-commit**:1.0	Not compliant.
4741, 6241	urn:ietf:params:netconf:capability:**rollback-on-error**:1.0	Compliance dependent on commit behavior. For more information, see Section 11.3 Transaction Commit on page 104.
4741, 6241	urn:ietf:params:netconf:capability:**validate**:1.0	Not compliant.(3)
4741, 6241	urn:ietf:params:netconf:capability:**startup**:1.0	Compliant. Supported operations: • <copy-config> • <delete-config>
4741, 6241	urn:ietf:params:netconf:capability:**url**:1.0	Not compliant.
4741, 6241	urn:ietf:params:netconf:capability:**xpath**:1.0	Not compliant.
5717	urn:ietf:params:netconf:capability:**partial-lock**:1.0	Not compliant.
5277	urn:ietf:params:netconf:capability:**notification**:1.0	Compliant. Supported operation: • <create-subscription>
5277	urn:ietf:params:netconf:capability:**interleave**:1.0	Not compliant.
6243	urn:ietf:params:netconf:capability:**with-defaults**:1.0	Not compliant.
Not standard	urn:ericsson:com:netconf:**notification**:1.0	COM NETCONF-specific extension.
Not standard	urn:ericsson:com:netconf:**action**:1.0	COM NETCONF-specific extension. Supported operation: • <action>
Not standard	urn:com:ericsson:ebase:0.1.0	COM NETCONF-specific extension.
Not standard	urn:com:ericsson:ebase:1.1.0	COM NETCONF-specific extension.
Not standard	urn:com:ericsson:ebase:1.2.0	COM NETCONF-specific extension.
Not standard	urn:ericsson:com:netconf:heartbeat:1.0	COM NETCONF-specific extension.
Not standard	urn:com:ericsson:netconf:operation:1.0	COM NETCONF-specific extension.

(1) Parameter <error-option>, if present, can only be set to rollback-on-error. Sending an <edit-config> operation with a different <error-option>, results in an <rpc-error> reply and the ending of the session
(2) Automatic locking is supported, see Section 11.2 Locking on page 103
(3) Automatic validation is supported, see Section 11.2 Locking on page 103.

Not supported operation requests, such as `<lock>`, `<unlock>`, `<commit>`, `<discard-changes>`, and `<validate>`, are replied with the standard message `operation-not-supported`.

For a section-by-section summary on RFC compliance, see Section 15 on page 125.

4 NETCONF Session

Both NETCONF over SSH and TLS can be configured to use other ports than the default port. For more information, refer to section *Port Allocations for CLI and NETCONF* in *COM Application Developer's Guide* and *Configuring SSH Daemons* in *COM SW Installation Base*.

In the examples, the default values for the ports are used. If other values are configured, then those values must be used when following the examples.

Multiple sessions can exist in parallel.

4.1 Start Session over SSH

Start a NETCONF Session over SSH (Deprecated due to COM SSHD management feature introduced)

Note: INSTRUCTION FOR DOCUMENT REUSE: Use this section only if the deprecated method is supported by the product.

To start a NETCONF session over SSH:

1. Start an SSH session with the following options:
   - **host** – O&M IP address or hostname of the Managed Element.
   - **port number** – TCP port `830` is default.

     Note: INSTRUCTION FOR DOCUMENT REUSE: Add product-specific port number. If the port number can be changed by the end user in public interfaces, then describe that procedure.
   - **subsystem** – Use subsystem `netconf`.
   - **user name** – Name of a valid user account.
   - **password** – Password for a valid user account.

Note: INSTRUCTION FOR DOCUMENT REUSE: Add product-specific options and default values.

The following is an example of logon with an OpenSSH client:

```
ssh <user>@<target_host> -p 830 -t -s netconf
```

Example 1 Logon with an OpenSSH Client

Start a NETCONF Session over SSH

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Use this section only if the COM SSHD management feature is supported by the product and it is manually enabled. The default state of SSH management feature in COM is disabled (false).

For more information on COM SSHD Management, refer to *COM System Architecture Description*.

To start a NETCONF session over SSH:

1. Start SSH session with the following options:
   - **host** – O&M IP address or hostname of the Managed Element.
   - **port number** – TCP port 830 is default.
   - **user name** – Name of a valid user account.
   - **password** – Password for a valid user account.

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Add product-specific options and default values.

Root user access is denied.

The following is an example of logon with an OpenSSH client:

```
ssh <user>@<target_host> -p 830
```

Example 2 Logon with an OpenSSH Client

The following is the same for both previous cases.

Note: If no message is transferred between the peers and the connection is lost because of network router or firewall change, peers can still consider a session active, as the underlying socket is still open.

To avoid this problem, use TCP keep alive feature that notifies peers on connection loss. Keep alive is an optional TCP feature, it is disabled by default and it is controlled by the following parameters:

- **Keepalive time** is the duration between two keepalive transmissions in idle condition. In most implementation, it defaults to 7200 seconds and its recommended value is 300 seconds.
- **Keepalive interval** is the duration between two successive keepalive retransmissions, if acknowledgment to the previous keepalive transmission is not received. Recommended value is 75 seconds
- **Keepalive retry** is the number of retransmissions to be carried out before declaring that remote end is not available. Recommended value is 9.

For details on how to change the keepalive settings, refer to the documentation of your SSH client or operating system.

Successful Session Start

When the session is established, peers must initiate the capabilities exchange, see Section 5.1.1 Hello Message on page 16.

Unsuccessful Session Start

The session start can fail. For examples of error messages, see Table 3.

Table 3 Session Start Error Messages

Error Message	Recommended Action
`SSH session timeout`	Check the IP address and port accessibility.
`Invalid Credentials`	Get a valid pair of user account and password from the System Administrator.
`Connection to COM failed (<socket_error>)`	Check if the NETCONF service is running. For details, see the socket error messages.

4.2 Start Session over TLS

To start a NETCONF session over TLS:

1. Start a TLS connection with the following options:

   - **host** – O&M IP address or hostname of the Managed Element.
   - **port number** – TCP port `6513` is default.

     Note: **INSTRUCTION FOR DOCUMENT REUSE:** Add product-specific port number. If the port number can be changed by the end user in public interfaces, then describe that procedure.

Note: The client certificate contains the user identity in field `subjectAltName`.

INSTRUCTION FOR DOCUMENT REUSE: Add product-specific certificate handling information that is relevant for the end user.

Successful Session Start

When the session is established, peers must initiate the capabilities exchange, see Section 5.1.1 Hello Message on page 16.

Unsuccessful Session Start

For error messages, see the product-specific documentation.

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Add link to product-specific document on error handling.

4.3 Session Inactivity Timer

A session is automatically ended without any warning when the predefined time has passed without activity. Activity in a session means any operation that results in data exchange between the client and server.

The default inactivity timer value is 5 minutes.

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Add product-specific time-out value.

A session that has an active notification subscription is not closed by the inactivity timer mechanism.

4.4 Session End

A NETCONF session is closed in the following cases:

- As a result of `<close-session>` or `<kill-session>` operations
- When the NETCONF session inactivity timer expires

- When the SSH session is closed by the client (**Ctrl+C** or **Ctrl+D**)
- When the SSH session is closed because of connection loss.
- All the ongoing TLS over NETCONF connections is ended if the `administrativeState` attribute of the`NetconfTls` MO is being set to `LOCKED` and is committed.

5 NETCONF RPC Messages

NETCONF messages are encoded in XML and exchanged on top of a simple RPC-based communication model with the following elements:

- NETCONF peers send `hello` messages to exchange capability information, see Section 5.1.1 Hello Message on page 16.
- NETCONF client sends `<rpc>` message to the session as operation request that contains `message-id`, which is an integer monotonically increased during the lifetime of the NETCONF session. For details on NETCONF operation request messages, see Section 7 on page 31.
- As a response to the operation request, NETCONF session sends a single `<rpc-reply>` message that contains the message identifier from the request and the following elements, depending on the message type:
  - On operation success, the `<rpc-reply>` element contains either an `<ok>` element, see Section 5.2 OK Message on page 18 or operation-specific elements according to the operation return value definition, see Section 7 on page 31.
  - Otherwise, the `<rpc-reply>` element contains `<rcp-error>` element if the operation is not completed without error, see Section 5.3 Error Message on page 18.
- NETCONF session sends `<notification>` messages to the subscribed clients if an event has occurred that complies to the notification trigger condition or if the client requests notification replay, see Section 8 on page 79.

Message `]]>]]>` termination sequence is used to provide message framing.

5.1 ebase Numbering

The ebase version is numbered according to: ebase:X.Y.Z, as follows:

- X is stepped when the new ebase is not backward compatible with the current ebase version regarding behavior or NETCONF expression.
- Y is stepped for functional growth that is backward compatible compared to the current X number or NETCONF syntax, that is, support for NETCONF expressions that was not previously supported is added. For example, now `<ok>` can be returned instead of error for the previously not supported expression.
- Z is stepped when an error is corrected in a backward compatible way compared to the current ebase.

5.1.1 Hello Message

To exchange information on the capabilities supported by the peers (both client and server), peers must send a `<hello>` message simultaneously, without waiting for the other peer.

As shown in Example 3, message `<hello>` sent by the server contains the list of supported capabilities and `<session-id>`, which is an integer increased by new session starts.

```
Server sends::
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <capabilities>
       <capability>urn:ietf:params:netconf:base:1.0</capability>
       <capability>urn:com:ericsson:ebase:0.1.0</capability>
       <capability>urn:com:ericsson:ebase:1.1.0</capability>
       <capability>urn:com:ericsson:ebase:1.2.0</capability>
       <capability>urn:ietf:params:netconf:capability:writable-running:1.0
       </capability>
       <capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0
       </capability>
       <capability>urn:ietf:params:netconf:capability:notification:1.0
       </capability>
       <capability>urn:ericsson:com:netconf:action:1.0</capability>
       <capability>urn:ericsson:com:netconf:heartbeat:1.0</capability>
       <capability>urn:com:ericsson:netconf:operation:1.0</capability>
       <capability>urn:ietf:params:netconf:capability:startup:1.0</capability>
   </capabilities>
   <session-id>1</session-id>
</hello>
]]>]]>
```

Example 3 Hello Message Sent by Server

Note: Each peer must send at least the base NETCONF capability in its `hello` message. The session is closed if the base capability is missing in the client hello message.

As shown in Example 4, message `<hello>` sent by the client must not contain `<session-id>` but the capability list with at least the `:base` capability.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <capabilities>
      <capability>urn:ietf:params:netconf:base:1.0</capability>
   </capabilities>
</hello>]]>]]>
```

Example 4 Hello Message Sent by Client

The session is closed without any error indication by either of the peers if the capability exchange fails, that is, if message `<hello>` is malformed or the advertised capabilities are insufficient.

After a successful capability exchange, the peers are ready to exchange any message that is defined for the capabilities.

As a limitation, the NETCONF server does not advertise the list of supported information models, thus the only way to get information on the models is by retrieving the instances of `Schema` by operation `<get>`.

5.1.2 ebase Capability Identifier

In line with the RFC, the highest common capability version from the client and server hello message determines the server behavior, as follows:

- If no common ebase capability is present, the default commit behavior of the SSH port is applied. For more information about setting the default port behavior, refer to sections *NETCONF TLS Configuration File* and *Port Allocations for CLI and NETCONF* in *COM Application Developer's Guide* and section *Configuring SSH Daemons* in *COM SW Installation Base*.
- If `urn:com:ericsson:ebase:0.1.0` is the highest common capability version, the changes are committed at the `<close-session>` request.
- If `urn:com:ericsson:ebase:1.1.0` is the highest common capability version, the changes are committed after each `<edit-config>` request.
- If `urn:com:ericsson:ebase:1.2.0` is the highest common capability version, behavior is set as per RFC compliance latest, see Section 14 on page 123.

In Example 5, the highest common capability is `urn:com:ericsson:ebase:0.1.0`, so in that session the changes done by `<edit-config>` requests are committed when the `<close-session>` arrives.

```
Server sends::
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
     <capability>urn:ietf:params:netconf:base:1.0</capability>
     <capability>urn:com:ericsson:ebase:0.1.0</capability>
     <capability>urn:com:ericsson:ebase:1.1.0</capability>
     <capability>urn:ietf:params:netconf:capability:writable-running:1.0
   </capability>
     <capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0
     </capability>
     <capability>urn:ietf:params:netconf:capability:notification:1.0
     </capability>
  </capabilities>
  <session-id>1</session-id>
</hello>]]>]]>

Client sends::
<?xml version="1.0" encoding="UTF-8"?>
   <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <capabilities>
      <capability>urn:ietf:params:netconf:base:1.0</capability>
      <capability>urn:com:ericsson:ebase:0.1.0</capability>
   </capabilities>
</hello>]]>]]>
```

Example 5 Capability Identifiers

In this case, the highest capability between peers is `urn:com:ericsson:ebase:0.1.0` and this triggers the transaction commit based on the `<close-session>` request.

5.2 OK Message

Operation completion without error is indicated by the `OK` message if the operation has no return value, as shown in Example 6.

```
<?xml version="1.0" encoding="UTF-8"?>
  <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     message-id="101">
    <ok/>
   </rpc-reply>
]]>]]>
```

Example 6 OK Message

Examples for operations without return value are `<close-session>` and `<kill-session>`.

5.3 Error Message

Any incomplete termination of the operation is indicated by a standard error message, see Example 7 through Example 11.

Element `<error-message>` provides COM NETCONF-specific details on the error.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply>
   <rpc-error>
      <error-type>protocol</error-type>
      <error-tag>operation-failed</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Malformed XML!!!</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 7 Error Message for Malformed XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>data-missing</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">MO: ManagedElement=NodeName,
      SystemFunctions=1,SysM=1,Snmp=3 is not available (no instance).
      </error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 8 Error Message for MO Instance Not Available

The error message returned to the MO creation request on a `systemCreated` MO is shown in Example 9.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>resource-denied</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Can not instantiate
      "ManagedElement=2". MO class ManagedElement, is system created
      </error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 9 Error Message for System-Created MOs

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>unknown-attribute</error-tag>
      <error-severity>error</error-severity>
      <error-info>
         <bad-attribute>rwattr2</bad-attribute>
         <bad-element>XThing</bad-element>
      </error-info>
      <error-message xml:lang="en">Invalid value s for an
       integer type attribute rwattr2</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 10 Error Message for Invalid Integer Value

The error message for setting another value than `Merge`, `Replace`, or `None` with `default-operation` is shown in Example 11.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="1">
   <rpc-error>
      <error-type>protocol</error-type>
      <error-tag>operation-not-supported</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en"> Default Operation
      Tag not supported with operation create </error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 11 Error Message for Setting Create As Default-Operation

6 NETCONF Data Model

This section describes the COM NETCONF XML format in terms of layout, containment, keying, lookup, replacement, management of the data and its relationship to the information models, as well as any other constraints imposed by the data model.

As a NETCONF principle, the format of configuration and state data in request messages is identical to the one in response messages, unless otherwise stated.

6.1 Model Definition

COM NETCONF XML elements represent MO instances of Managed Object Classes (MOCs), MO attributes, and MO actions as defined by the information models that are described in the MOM documentation.

The NETCONF client can request information on the supported model names, base model names, versions, and model options by a `<get>` operation on the `Schema` MOs defined in the `ECIM_SysM` model.

Information on model version, revision, and release is not available in COM NETCONF XML messages, because only one version of a model is supported at a time.

6.2 Namespaces

The XML namespace is defined for each model in Uniform Resource Name (URN) format with prefix `urn:com:ericsson:ecim:`, for example, as follows:

- `urn:com:ericsson:ecim:ECIM_Top`
- `urn:com:ericsson:ecim:ComTop`
- `urn:com:ericsson:ecim:ComFm`

The XML namespace, defined by XML attribute `xmlns`, is present in the NETCONF message sent by the server, for example, `<Fm xmlns="urn:com:ericsson:ecim:ComFm">`.

NETCONF messages sent by the client can contain namespace. However, they are ignored by the server, as the MO DN by itself identifies the MO instance and its class in a unique way within the scope of the Managed Element.

6.3 Configuration and State Data

Configuration Data

The configuration data is a set of data that is required to transform a system from its initial default state into its current state. This set of data is writable by the NETCONF client, as follows:

- An attribute is configurational data if it is not state data.
- An MO instance is configurational data if it is not state data.

State Data

The state data is additional data on a system that is not configuration data, such as read-only status information and collected statistics.

- An attribute is considered as state data in the following situations:
  - If It is defined as `readOnly`.
  - If it is key, and it is in an MO that has a relationship defined with its parent MO with property `canCreate` as `false` (or in case of legacy models the MO is `systemCreated`).
  - If it is restricted, and it is in an MO that has a relationship defined with its parent MO with property `canCreate` as `false` (or in case of legacy models the MO is `systemCreated`).
- An MO instance is state data if all the attributes are state data.

6.4 Distinguished Names

MO instances are identified by their Local Distinguished Names (LDNs), that is, the first element of the DN is `<ManagedElement>`.

A DN is represented as an XML element with a MOC name that contains an XML element with the MOC key attribute and value.

Class XML elements contain subordinate class elements according to their model-defined parent-child (MO containment) relationships.

DN `ManagedElement=NodeName` is represented as shown in Example 12.

```
<ManagedElement>
   <managedElementId>NodeName</managedElementId>
</ManagedElement>
```

Example 12 DN of ManagedElement

DN `ManagedElement=NodeName,SystemFunctions=1,Fm=1` is represented as shown in Example 13.

```
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
   <managedElementId>NodeName</managedElementId>
   <SystemFunctions>
      <systemFunctionsId>1</systemFunctionsId>
      <Fm xmlns="urn:com:ericsson:ecim:ComFm">
         <fmId>1</fmId>
      </Fm>
   </SystemFunctions>
</ManagedElement>
```

Example 13 DN of Fm in Request and Response Messages

As namespaces are optional in request messages, the form shown in Example 14 is also valid.

```
<ManagedElement>
   <managedElementId>NodeName</managedElementId>
   <SystemFunctions>
      <systemFunctionsId>1</systemFunctionsId>
      <Fm>
         <fmId>1</fmId>
      </Fm>
   </SystemFunctions>
</ManagedElement>
```

Example 14 DN of Fm in Request Messages without Namespace

A DN is subject to the 3GPP standard for MO Name convention. The allowed characters in a key value that make up part of the DN in a NETCONF message are therefore limited by this standard. Refer to 3GPP standard for MO Name convention, 3GPP TS 32.300 V9.0.0, for a full list of illegal characters. Illegal characters can be used if they are provided in their hexadecimal representation and then is not translated to their corresponding character. Legal characters in hexadecimal form are translated to their corresponding character.

Using the illegal ‘‘+’’ character in the key value of a DN, shown in Example 15:

```
<ManagedElement>
   <managedElementId>NodeName</managedElementId>
   <SystemFunctions>
      <systemFunctionsId>1</systemFunctionsId>
      <MoX>
         <moXId>keyWithAPlusCharacter\2B</moXId>
      </MoX>
   </SystemFunctions>
</ManagedElement>
```

Example 15 Using the Illegal + Character

6.5 MO Attributes

XML representation of an MO attribute is a child element of its encapsulating MO. In the operation response messages, the attributes are represented in the same order in the MO as defined in the model. In the operation request messages, the attributes can be represented in any order within the MO.

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Add information on product specific ordering, if different from the order of the one in model definition.

6.5.1 MO Key-Attribute

The Key-attribute modeling element is a MOC attribute. Key-attributes are represented as XML elements; first letter in lower-case, followed by the string “Id”. The Key-attribute value is provided between the opening and closing elements, for example: `<managedElementId><some value></managedElementId>`.

A Key-attribute can be any of the following data types:

- `EcimDerivedString`
- `EcimDerivedInteger`
- `EcimEnumeration`

A Key-attribute is a single-valued string attribute holding the RDN value of the MO instance it is contained in. The RDN value space is defined in 3GPP standard 32.300 version 9.0.0.

When a data type of the Key-attribute is an `EcimEnumeration`, the name part of its `EcimEnumerationLiterals` defines the allowed RDN values.

DN `ManagedElement=NodeName,NCTest=1,XThing=LOCKED` is represented as shown in Example 16.

```
ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
 <managedElementId>NodeName</managedElementId>
 <NCTest>
    <nCTestId>1</nCTestId>
    <XThing xc:operation="create">
      <xThingId>LOCKED</xThingId>
    </XThing>
 </NCTest>
</ManagedElement>
```

Example 16 DN of XThing in Request Messages

When a data type of the Key-attribute is an `EcimDerivedInteger`, the character string representing the integer value is used as RDN.

DN `ManagedElement=NodeName,NCTest=1,KThing=1` is represented as shown in Example 17.

```
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
 <managedElementId>NodeName</managedElementId>
 <NCTest>
    <nCTestId>1</nCTestId>
    <KThing xc:operation="create">
      <kThingId>1</kThingId>
    </KThing>
 </NCTest>
</ManagedElement>
```

Example 17 DN of KThing in Request Message

Table 4 indicates the `DerivedInteger` type input and the NETCONF interpretation for some special cases.

Table 4 DerivedInteger Type Input and NETCONF Interpretation

Input (DerivedInteger)	Interpretation
+1	1
+000000001	1
-1	-1
-00000000000000000001	-1
+000000000000000001	1
+00000000000000000	0
-0000000000000000	0

Example 18, Example 19, and Example 20 indicate the negative responses for the key-attributes of data types enumeration and Int.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
  <rpc-error>
     <error-type>application</error-type>
     <error-tag>invalid-value</error-tag>
     <error-severity>error</error-severity>
     <error-info>
        <bad-element>xThingId</bad-element>
     </error-info>
     <error-message xml:lang="en">Invalid value 1 for attribute
      xThingId</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

Example 18 Error Message for Invalid KeyAttribute Value of EcimEnumeration

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
  <rpc-error>
     <error-type>application</error-type>
     <error-tag>invalid-value</error-tag>
     <error-severity>error</error-severity>
     <error-info>
        <bad-element>kThingId </bad-element>
     </error-info>
     <error-message xml:lang="en">Invalid value hiiii for an integer type
      attribute kThingId</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

Example 19 Error Message for Invalid KeyAttribute Value of EcimDerivedInteger

Example 20 shows that an error message for integer value is not in the allowed range, that is, specified in the model as range/minimum and range/maximum.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
  <rpc-error>
     <error-type>application</error-type>
     <error-tag>invalid-value</error-tag>
     <error-severity>error</error-severity>
     <error-info>
        <bad-element>aThingId</bad-element>
     </error-info>
     <error-message xml:lang="en">Value 45 is out of range for attribute
      aThingId expected values from [0,40]</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

Example 20 Error Message for Integer Value Not Allowed Range

6.5.2 Single-Valued Attributes

Single-valued attributes are represented as XML elements, which are named after the attribute name. The attribute value is provided between the opening and closing elements, for example, *<userLabel><some value></userLabel>*, as shown in Example 21 and Example 22.

The values are represented as follows:

- Boolean values are represented as `true` or `false`, as shown in attribute `isMibWritable`.
- Integer values are represented as shown in attributes `snmpTargetV2CId`, `informRetryCount`, `port`, and `informTimeout`.
- String values are represented as UTF-8 values as shown in attribute `community` and `address`. XML escaping (`<` and `&#x41;`) is supported.
- Enum values are represented as strings, by the enumeration member name as shown in attributes `transportMethod` and `administrativeState`.
- `moRef` value is represented as LDN of the referred MO, for example, *<fileGroup>*`ManagedElement=NodeName,SystemFunctions=1,FileM=1,LogicalFs=1,FileGroup=1`*</fileGroup>*.
- Double value is represented as shown in attribute `distanceFactor`. The double datatype can be presented both in canonical form and scientific notation representation.

Example 21 shows how a single-valued attribute (`administrativeState`) is represented in the context of the DN (MOC and key attributes).

```
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
   <managedElementId>Node</managedElementId>
   <SystemFunctions>
      <systemFunctionsId>1</systemFunctionsId>
      <SysM xmlns="urn:com:ericsson:ecim:ComSysM">
         <sysMId>1</sysMId>
         <Snmp xmlns="urn:com:ericsson:ecim:ComSnmp">
            <snmpId>1</snmpId>
            <SnmpTargetV2C>
               <snmpTargetV2CId>1</snmpTargetV2CId>
               <administrativeState>UNLOCKED</administrativeState>
               <distanceFactor>1.5</distanceFactor>
            </SnmpTargetV2C>
         </Snmp>
      </SysM>
   </SystemFunctions>
</ManagedElement>
```

Example 21 Single-Valued Attribute

```
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
   <managedElementId>NodeName</managedElementId>
   <SystemFunctions>
      <systemFunctionsId>1</systemFunctionsId>
      <SysM xmlns="urn:com:ericsson:ecim:ComSysM">
         <sysMId>1</sysMId>
         <Snmp xmlns="urn:com:ericsson:ecim:ComSnmp">
            <snmpId>1</snmpId>
            <SnmpTargetV2C>
               <snmpTargetV2CId>1</snmpTargetV2CId>
               <community>private</community>
               <informRetryCount>1</informRetryCount>
               <address>127.0.0.1</address>
               <port>162</port>
               <informTimeout>300</informTimeout>
               <transportMethod>TRAP</transportMethod>
               <isMibWritable>true</isMibWritable>
               <administrativeState>UNLOCKED</administrativeState>
            </SnmpTargetV2C>
         </Snmp>
      </SysM>
   </SystemFunctions>
</ManagedElement>
```

Example 22 Multiple Single-Valued Attributes

An attribute that is defined as optional or nillable and has no value assigned is represented as follows:

- The attribute is not present in the `<get>` reply message if the `<get>` target is the container MO of the attribute or any of its parent MO.
- A message with `unset="true"` is returned if the `<get>` target is the attribute itself. For details, see Example 23.

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
         <managedElementId>NodeName</managedElementId>
         <userLabel unset="true"></userLabel>
      </ManagedElement>
   </data>
</rpc-reply>
]]>]]>
```

Example 23 Attribute without Value in Response Messages

6.5.3 Sequences

Attributes defined as sequence (multi-valued attributes) are represented as multiple occurrences of their single-valued attributes.

Attributes of multi-value type are expressed by repeating the same attribute XML tag with each value that it contains. Values in a multi-value are not ordered, that is, the order of the values when read out cannot be the same as when it was set.

A multi-value can be of type struct, that is containing more than one struct. A struct can contain multi-value members but not of struct type.

6.5.4 Structures

Attributes defined as struct are represented as XML elements that enclose members attributes. If the attribute is defined as structure, the struct XML attribute (`agentAddress struct="HostAndPort"`) is always present in the messages returned by the server, see Example 24, but it is not required to be present in messages sent by the client. The value of the struct XML attribute is the structure data type name.

Structure members attributes are represented in the same way as attributes of the same data type. A limitation is that a structure member cannot be of struct type.

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Add information on product specific ordering, if different from the order of the one in model definition.

```
<agentAddress struct="HostAndPort">
   <port>162</port>
   <host>10.20.30.40</host>
</agentAddress>
```

Example 24 Structure Attribute

Example 25 shows a structure attribute without a structure name.

```
<agentAddress>
   <port>162</port>
   <host>10.20.30.40</host>
</agentAddress>
```

Example 25 Structure Attribute without Structure Name

6.6 MO Instances

MO instances are represented as combined elements of the MO DNs and their attributes, see Example 26.

```
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
   <managedElementId>NodeName</managedElementId>
   <SystemFunctions>
      <systemFunctionsId>1</systemFunctionsId>
      <SysM xmlns="urn:com:ericsson:ecim:ComSysM">
         <sysMId>1</sysMId>
         <Snmp xmlns="urn:com:ericsson:ecim:ComSnmp">
            <snmpId>1</snmpId>
            <administrativeState>UNLOCKED</administrativeState>
            <agentAddress struct="HostAndPort"><port>161</port><host>10.0.0.1</host></agentAddress>
            <agentAddress struct="HostAndPort"><port>162</port><host>10.0.0.2</host></agentAddress>
            <agentAddress struct="HostAndPort"><port>163</port><host>10.0.0.3</host></agentAddress>
            <SnmpTargetV2C>
               <snmpTargetV2CId>1</snmpTargetV2CId>
               <community>private</community>
               <informRetryCount>1</informRetryCount>
               <address>127.0.0.1</address>
               <port>162</port>
               <informTimeout>300</informTimeout>
               <transportMethod>TRAP</transportMethod>
               <isMibWritable>true</isMibWritable>
               <administrativeState>UNLOCKED</administrativeState>
            </SnmpTargetV2C>
         </Snmp>
      </SysM>
   </SystemFunctions>
</ManagedElement>
```

Example 26 Snmp and Contained SnmpTargetV2C MO Instance

6.7 MO Actions

For description on MO actions, see Section 12 on page 107.

7 NETCONF Operations

This section provides descriptions and examples on the following NETCONF operations:

- `<get>`, see Section 7.1 <get> Operation on page 31
- `<get-config>`, see Section 7.2 <get-config> Operation on page 49
- `<edit-config>`, see Section 7.3 <edit–config> Operation on page 50
- `<action>`, see Section 7.4 <action> Operation on page 66
- `<create-subscription>`, see Section 7.5 <create-subscription> Operation on page 66
- `<close-session>`, see Section 7.6 <close-session> Operation on page 72
- `<kill-session>`, see Section 7.7 <kill-session> Operation on page 72
- `<copy-config>`, see Section 7.8 <copy-config> Operation on page 73
- `<delete-config>`, see Section 7.9 <delete-config> Operation on page 76

The following operations are supported according to the `urn:ietf:params:netconf:base:1.0` capability:

- `<lock>`
- `<unlock>`

Note: These two operation requests are always replied by the error message shown in Example 27.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="106">
   <rpc-error>
      <error-type>protocol</error-type>
      <error-tag>operation-failed</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Operation not supported</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 27 Response for Unsupported Operations

7.1 <get> Operation

Description

The `<get>` operation retrieves the configuration and state data from the “running” datastore.

Parameters

The operation has a single parameter, <filter>, that specifies the portion of the system configuration and state data to retrieve. If this parameter is not present, all the device configuration and state information is returned.

The <filter> can contain the <type> element. If the <type> element is present it must contain the subtree option, which is the only supported filter type. The <filter> parameter can contain filter expressions, see Section 7.1.3 <get> Filter on page 33. Notification stream discovery can be requested by a special filter construct, see Section 7.1.4 Request Event Stream Discovery on page 40.

The ‘‘xpath’’ option in <type> element is not valid, as the optional XPath capability is not supported.

Positive Response

If the operation is completed without error, an <rpc-reply> is sent. The <data> section contains the appropriate subset of configuration and state data in COM NETCONF XML format, see Section 6 on page 21.

Negative Response

An <rpc-error> element is included in the <rpc-reply> if the request cannot be completed for some reason.

As a special case, both <data> and <rpc-error> elements are present in the <rpc-reply> if the response sending is started but cannot be completed for some reason, see Section 7.1.1 Error during <get> Response Processing on page 32.

7.1.1 Error during <get> Response Processing

If the NETCONF server detects an error occurred during <get> response processing, partial response is provided. The <data> element in <rpc-reply> is closed, and an <rpc-error> message is returned, as shown in Example 28.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
         <managedElementId>1</managedElementId>
         <siteLocation unset="true"></siteLocation>
         <userLabel unset="true"></userLabel>
      </ManagedElement>
   </data>
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>resource-denied</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">get MO attribute returns error.
      Path: ManagedElement=1, attribute name: productIdentity</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 28 <get> Error

7.1.2 Error with Source Set to Startup

The `<get>` operation retrieves the configuration and runstate data from the running datastore. If the startup datastore is set as the source datastore, COM returns the RPC error message, as shown in Example 29.

```
<error-type>protocol</error-type>
<error-tag>operation-not-supported</error-tag>
<error-severity>error</error-severity>
<error-message xml:lang=\"en\">Failed to build operation</error-message>
```

Example 29 Response when Startup Datastore is Used

7.1.3 <get> Filter

Optionally, both `<get>` and `<get-config>` operation request messages can contain element `<filter>`. The only supported filter type is `subtree`, which is default value, that is, the filter expression is interpreted as filter `subtree` if attribute `type` is not present. Element `<filter>` can contain XML representations of DNs and MO attributes, see Section 6 on page 21.

The NETCONF protocol identifies the following five types of components that can be present in a subtree filter:

- Namespace Selection is not supported, as namespaces in the filter elements are ignored.
- Attribute Match Expressions are not supported, as no configuration data or state data is represented by COM NETCONF XML in XML attributes but in XML elements.
- Containment Nodes, Selection Nodes, and Content Match Nodes are supported, see Section 7.1.5 Support for Nodes on page 41.

7.1.3.1 No Filter

The complete content of the data store is returned if the filter element is not present, see Example 30.

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

Example 30 <get> Request without Filter

7.1.3.2 Empty Filter

A filter with an empty value matches no element as required by the standard, see Example 31 and Example 32.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
      <filter type="subtree">
      </filter>
   </get>
</rpc>]]>]]>
```

Example 31 <get> Request with Empty Filter

```
<?xml version="1.0" encoding="UTF-8"?>
   <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="123">
      <data>
      </data>
   </rpc-reply>
]]>]]>
```

Example 32 Reply Message

7.1.3.3 Filter on MO Class

MO instances of the specified class are returned if there is a selection node in the filter expression, as `Fm` in Example 33.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
      <filter type="subtree">
         <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
            <managedElementId>NodeName</managedElementId>
            <SystemFunctions>
               <systemFunctionsId>1</systemFunctionsId>
               <Fm/>
            </SystemFunctions>
         </ManagedElement>
      </filter>
   </get>
</rpc>]]>]]>
```

Example 33 Filtered <get> on Fm

`<ManagedElement>` is the root of the MO tree, that is, the whole MO subtree contained in the data store, is returned if parameter `<filter>` contains `<ManagedElement/>` or `<ManagedElement></ManagedElement>`, see Example 34.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
      <filter type="subtree">
         <ManagedElement/>
      </filter>
   </get>
</rpc>]]>]]>
```

Example 34 Filtered <get> on ManagedElement

7.1.3.4 DN Filter

All the attributes of an MO and its contained MO are returned if filter `<get>` contains the XML representation of the DN, see Example 35 and Example 37.

```
<rpc message-id="1"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
      <filter type="subtree">
         <ManagedElement>
            <managedElementId>NodeName</managedElementId>
            <SystemFunctions>
               <systemFunctionsId>1</systemFunctionsId>
               <Fm>
                  <fmId>1</fmId>
               </Fm>
            </SystemFunctions>
         </ManagedElement>
      </filter>
   </get>
</rpc>]]>]]>
```

Example 35 <get> Operation Request with DN Filter on Fm MO

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
         <managedElementId>NodeName</managedElementId>
         <SystemFunctions>
            <systemFunctionsId>1</systemFunctionsId>
            <Fm xmlns="urn:com:ericsson:ecim:ComFm">
               <fmId>1</fmId>
               <lastSequenceNo>2</lastSequenceNo>
               <sumCritical>1</sumCritical>
               <sumMajor>0</sumMajor>
               <sumMinor>1</sumMinor>
               <sumWarning>0</sumWarning>
               <totalActive>3</totalActive>
               <lastChanged>2012-11-15T00:26:46+01:00</lastChanged>
               <heartbeatInterval>60</heartbeatInterval>
               <FmAlarm>
                  <fmAlarmId>1</fmAlarmId>
                  <source>systemRecovery=</source>
                  <lastEventTime>2012-11-14T16:27:30+01:00</lastEventTime>
                  <sequenceNumber>1</sequenceNumber>
                  <activeSeverity>CRITICAL</activeSeverity>
                  <additionalText>BACKUP=backups/bb</additionalText>
                  <majorType>193</majorType>
                  <minorType>169</minorType>
                  <specificProblem>Zone Reloaded From Backup</specificProblem>
                  <eventType>EQUIPMENTALARM</eventType>
                  <probableCause>0</probableCause>
                  <additionalInfo></additionalInfo>
               </FmAlarm>
               <FmAlarm>
                  <fmAlarmId>2</fmAlarmId>
                  <source>FileSystemBackup=</source>
                  <lastEventTime>2012-11-14T16:56:46+01:00</lastEventTime>
                  <sequenceNumber>2</sequenceNumber>
                  <activeSeverity>MINOR</activeSeverity>
                  <additionalText>LastArchivingTime=: is not known</additionalText>
                  <majorType>193</majorType>
                  <minorType>176</minorType>
                  <specificProblem>IO, Archiving interval exceeded</specificProblem>
                  <eventType>QUALITYOFSERVICEALARM</eventType>
                  <probableCause>0</probableCause>
                  <additionalInfo></additionalInfo>
               </FmAlarm>
```

Example 36 <get> Operation Reply Message with Fm MO

```
               <FmAlarmModel>
                  <fmAlarmModelId>tsp</fmAlarmModelId>
                  <FmAlarmType>
                     <fmAlarmTypeId>IOArchivingIntervalExceeded</fmAlarmTypeId>
                     <majorType>193</majorType>
                     <minorType>176</minorType>
                     <moClasses>FileSystemBackup</moClasses>
                     <specificProblem>IO, Archiving interval exceeded</specificProblem>
                     <eventType>QUALITYOFSERVICEALARM</eventType>
                     <probableCause>0</probableCause>
                     <isStateful>true</isStateful>
                     <additionalText></additionalText>
                  </FmAlarmType>
                  <FmAlarmType>
                     <fmAlarmTypeId>SwitchBoardPowerFailure</fmAlarmTypeId>
                     <majorType>193</majorType>
                     <minorType>167</minorType>
                     <moClasses>SwitchBoard</moClasses>
                     <specificProblem>Switch Board, Power Failure</specificProblem>
                     <eventType>EQUIPMENTALARM</eventType>
                     <probableCause>58</probableCause>
                     <isStateful>true</isStateful>
                     <additionalText></additionalText>
                  </FmAlarmType>
                  <FmAlarmType>
                     <fmAlarmTypeId>ZoneReloadedFromBackup</fmAlarmTypeId>
                     <majorType>193</majorType>
                     <minorType>169</minorType>
                     <moClasses>systemRecovery</moClasses>
                     <specificProblem>Zone Reloaded From Backup</specificProblem>
                     <eventType>EQUIPMENTALARM</eventType>
                     <probableCause>0</probableCause>
                     <isStateful>true</isStateful>
                     <additionalText></additionalText>
                  </FmAlarmType>
               </FmAlarmModel>
            </Fm>
         </SystemFunctions>
      </ManagedElement>
   </data>
</rpc-reply>
]]>]]>
```

7.1.3.5 DN Filter on Multiple Branches

The DN filter can contain multiple combined DN elements by either of the ways shown in Example 37 and Example 38.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
      <filter type="subtree">
         <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
            <managedElementId>NodeName</managedElementId>
            <NCTest>
               <nCTestId>1</nCTestId>
               <XThing>
                  <xThingId>1</xThingId>
               </XThing>
               <YThing>
                  <yThingId>1</yThingId>
               </YThing>
            </NCTest>
         </ManagedElement>
      </filter>
   </get>
</rpc>]]>]]>
```

Example 37 <get> Operation Request with DN Filter on Multiple MOs

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-config>
      <source>
         <running/>
      </source>
      <filter>
         <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
            <managedElementId>NodeName</managedElementId>
            <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
               <nCTestId>1</nCTestId>
               <XThing>
                  <xThingId>1</xThingId>
               </XThing>
            </NCTest>
         </ManagedElement>
         <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
            <managedElementId>NodeName</managedElementId>
            <NCTest>
               <nCTestId>1</nCTestId>
               <YThing>
                  <yThingId>1</yThingId>
               </YThing>
            </NCTest>
         </ManagedElement>
      </filter>
   </get-config>
</rpc>]]>]]>
```

Example 38 <get> Operation Request with DN Filter on Multiple MOs

The operation reply contains both requested elements, as shown in Example 39.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
<data>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
       <managedElementId>NodeName</managedElementId>
       <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
         <nCTestId>1</nCTestId>
         <XThing>
           <xThingId>1</xThingId>
           <rwattr1>RW-One</rwattr1>
           <rwattr2>2</rwattr2>
           <XXThing>
             <xXThingId>1</xXThingId>
             <rwattr1>RW-One</rwattr1>
             <rwattr2>2</rwattr2>
           </XXThing>
         </XThing>
         <YThing>
           <yThingId>1</yThingId>
           <rwattr1>RW-One</rwattr1>
           <rwattr2>2</rwattr2>
        </YThing>
      </NCTest>
    </ManagedElement>
</data>
</rpc-reply>
```

Example 39 Multiple Requested Elements in Operation Response

7.1.3.6 Filtered <get> on Simple-Valued Attribute

Selected attributes are returned if filter `<get>` contains the XML representation of the attributes, see Example 40 and Example 41.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
     <source>
        <running/>
     </source>
     <filter type="subtree">
        <ManagedElement>
           <managedElementId>NodeName</managedElementId>
           <timeZone></timeZone>
           <siteLocation></siteLocation>
        </ManagedElement>
     </filter>
  </get-config>
</rpc>
]]>]]>
```

Example 40 <get> Operation Request with Attribute Filter

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
         <managedElementId>NodeName</managedElementId>
         <siteLocation>siteLocationValue</siteLocation>
         <timeZone>timeZoneValue</timeZone>
      </ManagedElement>
   </data>
</rpc-reply>
]]>]]>
```

Example 41 <get> Operation Reply with MO Attributes

The reply message, shown in Example 42, is returned if the specified attribute is defined as optional or nillable and has no value assigned.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
         <managedElementId>NodeName</managedElementId>
         <userLabel unset="true"></userLabel>
      </ManagedElement>
   </data>
</rpc-reply>
]]>]]>
```

Example 42 <get> Operation Reply with Attribute Value Not Set

The error message shown in Example 43 is returned if the specified attribute is not defined.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>unknown-attribute</error-tag>
      <error-severity>error</error-severity>
      <error-info>
         <bad-attribute>userLabelXXX</bad-attribute>
         <bad-element>ManagedElement</bad-element>
      </error-info>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 43 <get> Operation Reply with Attribute Not Defined

7.1.3.7 Filtered <get> on Struct Attribute

A selected attribute that is defined as struct is returned if filter `<get>` contains the XML representation of the attribute name of the struct, see Example 44 and Example 45. Filter on struct member attribute (such as `anInt64Attr` in Example 45) is not supported.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
<filter>
   <ManagedElement>
      <managedElementId>NodeName</ex:managedElementId>
      <NCTest>
         <nCTestId>1</nCTestId>
         <SimpleStruct>
            <simpleStructId>1</simpleStructId>
            <attrFileData/>
         </SimpleStruct>
      </NCTest>
   </ManagedElement>
</filter>
</get>
</rpc>]]>]]>
```

Example 44 <get> Operation Request with Attribute Filter

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ex="urn:com:ericsson:ecim:ComTop"
xmlns:ex2="urn:com:ericsson:ecim:ncmom1" message-id="1">
   <data>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
         <managedElementId>1</managedElementId>
         <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
            <nCTestId>1</nCTestId>
            <SimpleStruct>
               <simpleStructId>1</simpleStructId>
               <attrFileData struct="FileData">
                  <anInt64Attr>33</anInt64Attr>
                  <aStringAttr>test.jpg</aStringAttr>
                  <aBoolAttr>true</aBoolAttr>
                  <aBoolAttrNoDefault>false</aBoolAttrNoDefault>
                  <anInt32Attr>3</anInt32Attr>
                  <anInt32Attr>4</anInt32Attr>
               </attrFileData>
            </SimpleStruct>
         </NCTest>
      </ManagedElement>
   </data>
</rpc-reply>
```

Example 45 <get> Operation Reply with MO Attributes

7.1.4 Request Event Stream Discovery

As an RFC 5277 defined function, a NETCONF client can retrieve the list of supported event streams from a NETCONF server by operation `<get>` on `<streams>` element, see Example 46. Only the mandatory `NETCONF` stream is supported by COM NETCONF. The type attribute must be `subtree`, which is the only supported filter type.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
      <filter type="subtree">
         <netconf xmlns="urn:ietf:params:xml:ns:netmod:notification">
            <streams/>
         </netconf>
      </filter>
   </get>
</rpc>]]>]]>
```

Example 46 Requesting Event Stream Discovery

As shown in Example 47, the event stream discovery reply message contains the following information on the single supported stream:

- `name` – is `NETCONF`
- `description` – is `default NETCONF event stream`
- `replaySupport` – contains `true`
- `replayLogCreationTime` – indicates the date and time of the earliest available logged notification.
- `replayLogAgedTime` – indicates the date and time of the last notification aged out of the log. This XML element is present if notifications have been aged out of the log.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="101">
   <data>
      <netconf xmlns="urn:ietf:params:xml:ns:netmod:notification">
         <streams>
            <stream>
               <name>NETCONF</name>
               <description>default NETCONF event stream</description>
               <replaySupport>true</replaySupport>
               <replayLogCreationTime>2011-11-24T12:52:19+01:00
               </replayLogCreationTime>
               <replayLogAgedTime>2011-11-24T13:45:03+01:00
               </replayLogAgedTime>
            </stream>
         </streams>
      </netconf>
   </data>
</rpc-reply>
]]>]]>
```

Example 47 Event Stream Discovery Reply

Note: COM NETCONF does not support user-specified filters on the `<get>` operation.

7.1.5 Support for Nodes

NETCONF supports containment node, selection node, and content match node components in the subtree filter for both `<get>` and `<get-config>` operations.

7.1.5.1 Containment Node

A node that contains child elements within a subtree filter is called containment node. Each child element can be any type of node, including another containment node. For each containment node specified in a subtree filter, all data model instances that exactly match the specified namespaces, element hierarchy, and any attribute match expressions, are included in the filter output.

In Example 48 and Example 49, the `<ManagedElement>` element is a containment node.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
      <filter type="subtree">
         <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
              <managedElementId>1</managedElementId>
              <userLabel/>
            </ManagedElement>
      </filter>
   </get>
</rpc>]]>]]>
```

Example 48 Requesting Containment Node

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
     <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>1</managedElementId>
        <userLabel>UserLabelValue</userLabel>
     </ManagedElement>
  </data>
</rpc-reply>
]]>]]>
```

Example 49 Containment Node Reply

7.1.5.2 Selection Node

An empty leaf node within a filter is called a selection node and it represents an explicit selection filter on the underlying data model. Presence of any selection nodes within a set of sibling nodes cause the filter to select the specified subtrees and suppress automatic selection of the entire set of sibling nodes in the underlying data model. For filtering purposes, an empty leaf node can be declared either with an empty tag, for example, `<foo/>`, or with explicit start and end tags, for example, `<foo> </foo>`. Any white-space characters are ignored in this form.

In Example 50, the request `ManagedElement` is a containment node with content match, `NCTest` is a containment node, `Xthing` is an empty leaf node, which is called selection node. Whenever a selection node is present in another node, it means that the other not explicitly stated attributes or children to the ‘‘other’’ node is not returned, except for the ID attribute that is always returned. Attributes specified in a content match are always returned. In Example 50, the `managedElementId` is returned but no other attributes of `ManagedElement` are returned. If a containment node only contains content match nodes, then

all attributes and children to that node are returned. The trivial case is the id attribute match.

The filter is a request for all `XThing` instances under all `NCTest` instances under `ManagedElement` instances where `managedElementId = 1`, which happens to be the value of the DN id attribute for this MO. If more attributes are included under a `ManagedElement` in the filter, then they are evaluated as a logical **AND** that must be true for a filter match, as in Example 51. Any children to matching `XThing` are also returned.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
<filter type="subtree">
   <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
     <managedElementId>1</managedElementId>
     <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
        <XThing/>
     </NCTest>
   </ManagedElement>
</filter>
</get>
</rpc>]]>]]>
```

Example 50 Requesting Selection Node

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
       <managedElementId>1</managedElementId>
       <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
          <nCTestId>1</nCTestId>
          <XThing>
            <xThingId>1</xThingId>
            <roattr1>RO-One</roattr1>
            <roattr2>2</roattr2>
            <rwattr1>RW-One</rwattr1>
            <rwattr2>2</rwattr2>
            <XXThing>
               <xXThingId>1</xXThingId>
               <roattr1>RO-One</roattr1>
               <roattr2>2</roattr2>
               <rwattr1>RW-One</rwattr1>
               <rwattr2>2</rwattr2>
           </XXThing>
        </XThing>
     </NCTest>
   </ManagedElement>
</data>
</rpc-reply>]]>]]>
```

Example 51 Selection Node Reply

7.1.5.3 Content Match Node

A leaf node that contains simple content is called a content match node. It represents an exact-match filter on the leaf network element content and it is used to select some or all its sibling nodes to filter output.

The following constraints apply to content match nodes:

- A content match node must not contain nested elements.

- Multiple content match nodes, which are sibling nodes, are logically combined in an **AND** expression.
- Filtering of mixed content is not supported.
- Filtering of list content is not supported.
- Filtering of whitespace-only content is not supported.
- A content match node must contain non-whitespace characters. An empty element, for example, `<foo></foo>`, is interpreted as a selection node, for example, `<foo/>`.
- Leading and trailing whitespace characters are ignored. However, any whitespace characters within a block of text characters are not ignored or modified.

If all specified sibling content match nodes in a subtree filter expression are true, then the filter output nodes are selected in the following manner:

- Each content match node in the sibling set is included in the filter output.
- If any containment nodes are present in the sibling set, then they are processed further and included if any nested filter criteria are also met.
- If any selection nodes are present in the sibling set, then all them are included in the filter output.
- If any sibling nodes of the selection node are instance identifier components for a conceptual data structure, for example, a list key leaf, then they can also be included in the filter output.
- Otherwise, if there are no selection or containment nodes in the filter sibling set, all the nodes defined at this level in the underlying data model, and their subtrees (if any), are returned in the filter output.

If any of the sibling content match node tests are false, then no further filter processing is performed on that sibling set, and none of the sibling subtrees are selected by the filter, including the content match nodes.

In Example 52 and Example 53 a request for the subtree `XThing` where `xthingId=1` and `rwAttr2=1` under all `NCTest` instances under `ManagedElement` where `managedElementId=1`. Since no selection node is present in `XThing`, all its attributes and subtrees are returned.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
<filter type="subtree">
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
       <managedElementId>1</managedElementId>
       <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
         <XThing>
           <xThingId>1</xThingId>
           <rwattr1>RW-One</rwattr1>
         </XThing>
      </NCTest>
    </ManagedElement>
</filter>
</get>
</rpc>]]>]]>
```

Example 52 Requesting Content Match Node

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
<data>
  <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
    <managedElementId>1</managedElementId>
    <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
      <nCTestId>1</nCTestId>
      <XThing>
        <xThingId>1</xThingId>
        <roattr1>RO-One</roattr1>
        <roattr2>2</roattr2>
        <rwattr1>RW-One</rwattr1>
        <rwattr2>2</rwattr2>
        <XXThing>
          <xXThingId>1</xXThingId>
          <roattr1>RO-One</roattr1>
          <roattr2>2</roattr2>
          <rwattr1>RW-One</rwattr1>
          <rwattr2>2</rwattr2>
        </XXThing>
      </XThing>
    </NCTest>
  </ManagedElement>
</data>
</rpc-reply>
]]>]]>
```

Example 53 Content Match Node Reply

7.1.5.3.1 Error during Content Match Node Processing

If the content match specified in the subtree filtering is false, that is, it does not match with any of the subtree filtering, then the error is displayed as shown in Example 54.

```
<?xml version="1.0" encoding="UTF-8"?>
   <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
     <rpc-error>
       <error-type>application</error-type>
       <error-tag>data-missing</error-tag>
       <error-severity>error</error-severity>
       <error-message xml:lang="en">MO: ManagedElement=1,NCTest,XThing=1 is not
       available with requested content (no instance).</error-message>
     </rpc-error>
</rpc-reply>]]>]]>
```

Example 54 Error during Content Match Node Processing

7.1.6 Optional MO Instances Specification in <Get>/<Get-config> Request

Optional MO Instances specification in the `Get`/`Get-config` request.

The Example 55 shows the optional MO instances request.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
<filter type="subtree">
  <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
    <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
      <XThing>
        <XXThing/>
      </XThing>
    </NCTest>
  </ManagedElement>
</filter>
</get>
</rpc>]]>]]>
```

Example 55 Optional MO Instances Request

The Example 56 shows the optional MO instances replay.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
<data>
 <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
   <managedElementId>1</managedElementId>
   <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
     <nCTestId>1</nCTestId>
     <XThing>
       <xThingId>1</xThingId>
       <XXThing>
         <xXThingId>1</xXThingId>
         <roattr1>RO-One</roattr1>
         <roattr2>2</roattr2>
         <rwattr1>RW-One</rwattr1>
         <rwattr2>2</rwattr2>
       </XXThing>
     </XThing>
   </NCTest>
 </ManagedElement>
</data>
</rpc-reply>
]]>]]>
```

Example 56 Optional MO Instances Reply

Note: For a few types of `get` / `get-config`, NETCONF requests (requests without key-ID for the MO and this MO is a selection node at the last level in the request), memory footprint is allocated proportionately to the number of MO instances at that particular point in the MO tree during the execution of the NETCONF request.

For the `get` request in Example 57, memory footprint is proportionate to the number of instances of `XThing`.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
      <filter type="subtree">
         <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
         <managedElementId>1</managedElementId>
            <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
               <nCTestId>1</nCTestId>
               <XThing/>
            </NCTest>
         </ManagedElement>
      </filter>
   </get>
</rpc>]]>]]>
```

Example 57 get Request

7.1.7 <get> Operation with RFC Compliance Latest Mode

The `<get>` operation does not return any error if the requested MO does not exist.

If the `<get>` operation contains request for non-instantiated child MO, as shown in Example 58, then the response is generated with empty `<data>` tags, as show in Example 59. Here `xThingId=2` and `yThingId=2` are not instantiated, then the response is generated with empty `<data>` tags.

```
<?xml version="1.0" encoding="UTF-8"?>
   <rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <get>
         <filter type="subtree">
            <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
               <managedElementId>1</managedElementId>
               <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
                  <nCTestId>1</nCTestId>
                  <XThing >
                     <xThingId>2</xThingId>
                  </XThing>
                  <YThing >
                     <yThingId>2</yThingId>
                  </YThing>
               </NCTest>
          </ManagedElement>
      </filter>
   </get>
</rpc>
]]>]]>
```

Example 58 Request for Non-Instantiated Child MO

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

Example 59 Response with Empty <data> Tags

If the `<get>` operation requests non-instantiated MO, as shown in Example 60, then the response to the `<get>` operation displays empty `<data>` tags, as shown in the Example 59.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
   <get>
      <filter type="subtree">
        <ManagedElement>
           <managedElementId>1</managedElementId>
           <NCTest>
              <nCTestId>1</nCTestId>
              <Athing/>
              <Ggsn/>
           </NCTest>
         </ManagedElement>
      </filter>
   </get>
</rpc>
]]>]]>
```

Example 60 Request for Non-Instantiated MO

If the <get> request contains multiple MO requests, where some MOs have instances and other MOs have no instance, that has instances are displayed in the response to the get request, as shown in Example 61 and Example 62:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId></managedElementId>
        <NCTest>
          <nCTestId>1</nCTestId>
          <XThing >
            <xThingId>1</xThingId>
          </XThing>
          <YThing>
            <yThingId>2</yThingId>
          </YThing>
        </NCTest>
      </ManagedElement>
    </filter>
  </get>
</rpc>
]]>]]>
```

Example 61 Request Containing Instantiated and Non-Instantiated MOs

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
       <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
           <managedElementId>1</managedElementId>
           <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
               <nCTestId>1</nCTestId>
               <XThing>
                   <xThingId>1</xThingId>
                   <roattr1>RO-One</roattr1>
                   <roattr2>2</roattr2>
                   <rwattr1>RW-One</rwattr1>
                   <rwattr2>2</rwattr2>
                   <XXThing>
                       <xXThingId>1</xXThingId>
                       <roattr1>RO-One</roattr1>
                       <roattr2>2</roattr2>
                       <rwattr1>RW-One</rwattr1>
                       <rwattr2>2</rwattr2>
                   </XXThing>
               </XThing>
           </NCTest>
       </ManagedElement>
   </data>
</rpc-reply>
```

Example 62 Response Containing Instantiated MO

7.2 <get-config> Operation

The operation request and response messages of operation `<get-config>` are identical to those of operation `<get>` with the following exceptions:

- The request message of operation `<get-config>` contains an additional parameter `<source>` that identifies the configuration data store being queried. The source parameter must be `running`, see Example 63.
- The reply message of operation `<get-config>` contains configuration data but no state data. For definition of configuration and state data, see Section 6.3 Configuration and State Data on page 22.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
 <get-config>
   <source>
      <running/>
   </source>
   <filter type="subtree">
     <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>1</managedElementId>
        <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
            <nCTestId>1</nCTestId>
            <XThing>
               <xThingId>1</xThingId>
            </XThing>
       </NCTest>
     </ManagedElement>
  </filter>
</get-config>
</rpc>
]]>]]>
```

Example 63 <get-config> Operation Request Message

7.3 <edit–config> Operation

Description

The `<edit-config>` operation loads or changes all or part of a specified configuration in the running `datastore`.

Parameters

The `<edit-config>` operation can be requested with the following parameters:

target It must be running.

default-operation

The `default-operation` parameter is optional, but if provided, it must have one of the following values:

- `merge`: The configuration data in the `<config>` parameter is merged with the configuration at the corresponding level in the target `datastore`, as shown in Example 64.
- `replace`: The configuration data in the `<config>` parameter completely replaces the configuration in the target `datastore`.
- `none`: The target `datastore` is unaffected by the configuration in the `<config>` parameter, unless and until the incoming configuration data uses the `operation` attribute to request a different operation.

```
<?xml version="1.0" encoding="UTF-8"?>
   <rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <edit-config>
         <target>
            <running/>
        </target>
     <default-operation>merge</default-operation>
 <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ManagedElement>
   <managedElementId>NodeName</managedElementId>
   <SystemFunctions>
    <systemFunctionsId>1</systemFunctionsId>
    <SysM>
     <sysMId>1</sysMId>
     <NtpServer>
      <ntpServerId>1</ntpServerId>
      <serverAddress>127.0.0.1</serverAddress>
     </NtpServer>
     <NtpServer>
    </SysM>
   </SystemFunctions>
  </ManagedElement>
 </config>
 </edit-config>
</rpc>
]]>]]>
```

Example 64 Default-Operation Tag with Merge Option

error-option

This element can be present. However, the only supported behavior is “rollback–on–error”. If receiving an `<edit-config>` request containing `<error-option>` “stop–on–error” and “continue–on–error”, an error message is returned, which indicates that this `<error-option>` is not supported and the session is closed if the session commit behavior is “commit at `<close-session>`”, or else the session is continued.

config

This mandatory parameter identifies the target configuration. Target configuration consists of one or multiple MOs and attributes, that is, multiple MOs or MO subtrees can be changed with one `<edit-config>` operation.

Elements in the `<config>` subtree can contain `operation` attributes that identify the points in the configuration where the operation is to be performed. The `operation` attribute can appear in multiple elements throughout the `<config>` subtree. The operation attribute has one of the following values:

- `merge`: The configuration data identified by the element containing this attribute is merged with the configuration at the corresponding level in the configuration datastore identified by the `<target>` parameter. This is the default behavior. For examples, see Section 7.3.4 <edit–config> Operation with <merge> Option on page 58

- `create`: The configuration data identified by the element containing this attribute is added to the configuration if and only if the configuration data does not exist in the configuration datastore. If the configuration data exists, an `<rpc-error>` element is returned with an `<error-tag>` value of `data-exists`. For examples, see Section 7.3.1 <edit–config> Operation with <create> Option on page 53

- `delete`: The configuration data identified by the element containing this attribute is deleted from the configuration if and only if the configuration data currently exists in the configuration datastore. If the configuration data does not exist, an `<rpc-error>` element is returned with an `<error-tag>` value of `data-missing`. For examples, see Section 7.3.2 <edit–config> Operation with <delete> Option on page 55

- `replace`: The configuration data identified by the element containing this attribute replaces any related configuration in the configuration data store identified by the `<target>` parameter. If no such configuration data exists in the configuration data store, it is created. Unlike a `<copy-config>` operation, which replaces the entire target configuration, only the configuration present in the `<config>` parameter is affected. For examples, see Section 7.3.5 <edit–config> Operation with <replace> Option on page 64

- `remove`: The configuration data identified by the element containing this attribute is deleted from the configuration if the configuration data currently exists in the configuration data store. If the configuration data does not exist, the `remove` operation is silently ignored by the server. This option is not supported, as a limitation to RFC 6241.

If the `operation` attribute is not present, the `<edit-config>` operation is completed according to the `merge` option.

Note:

The `<test-option>` parameter must not be present, as the `:validate` capability is not supported.

Positive Response

If the <edit-config> operation is completed without error, an <ok> message is returned.

- If the session commit behavior is "commit at <close-session>", the affected MO instances are locked and the configuration change is added to the session view of the running configuration. The configuration change is performed at transaction commit when the client issues a <close-session> operation.
- If the session commit behavior is "commit after <edit-config>", the changes are committed into the configuration and visible from any session.

Negative Response

An <rpc-error> response is sent if the request cannot be completed for any of the following reasons:

- Validation rules derived from the model properties are violated, see Section 9 on page 81.
- Implementation-specific validation rule defined for the MO instance or to the MO attribute is violated.
- The MO attribute, the MO instance, or its parent MO is locked. For details on locking, see Section 11.2 Locking on page 103.
- Internal processing error occurred, see Section 5.3 Error Message on page 18.

All the changes issued in the failing <edit-config> request are rolled back. The effect on the previous <edit-config> operations in the same session depends on the commit behavior, as follows:

- If the session commit behavior is "commit at <close-session>", after the <rpc-error> response, the NETCONF server closes the session. Changes done in the same session by previous successful <edit-config> operations get rolled back as well.
- If the session commit behavior is "commit after <edit-config>", the session remains open. The changes from previous <edit-config> operations in the same session are already committed into the database.

For more information, see Section 11.3 Transaction Commit on page 104.

7.3.1 <edit–config> Operation with <create> Option

The <create> operation results in MO instance creation and creation of attributes is not supported. Use "merge" option to assign values to an unset attribute for an existing MO.

7.3.1.1 Create MO Instances

The <edit-config> operation triggers an MO instance creation request if the xc:operation="create" attribute is present in the class element of an MO instance.

For an example on creating two NtpServer MOs, see Example 65.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement>
            <managedElementId>NodeName</managedElementId>
            <SystemFunctions>
               <systemFunctionsId>1</systemFunctionsId>
               <SysM>
                  <sysMId>1</sysMId>
                  <NtpServer xc:operation="create">
                     <ntpServerId>1</ntpServerId>
                     <serverAddress>127.0.0.1</serverAddress>
                  </NtpServer>
                  <NtpServer xc:operation="create">
                     <ntpServerId>2</ntpServerId>
                     <serverAddress>127.0.0.1</serverAddress>
                  </NtpServer>
               </SysM>
            </SystemFunctions>
         </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>
```

Example 65 Creating MO Instances

7.3.1.2 Create Operation with RFC Compliance Latest Mode

The <edit-config> operation with create option provided at parent MO is inherited to the child MOs if any. This applies even if the <default-operation> contains “none” option.

The <edit-config> operation as shown in Example 66 is a request to create an MO. Here along with creation of AThing parent MO, even AAThing child MO is created, even if <default-operation> contains “none”.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
  <target>
  <running/>
  </target>
  <default-operation>none</default-operation>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>1</managedElementId>
        <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
          <nCTestId>1</nCTestId>
          <AThing xc:operation="create">
            <aThingId>2</aThingId>
            <rwattr2>27</rwattr2>
          <AAThing>
            <aAThingId>2</aAThingId>
            <rwattr2>52</rwattr2>
          </AAThing>
        </AThing>
        </NCTest>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>]]>]]>
```

Example 66 Request for Create Operation with Default-Operation Set to None

7.3.2 <edit–config> Operation with <delete> Option

Depending on the operation request and on the current configuration, the `<delete>` operation results in MO instance or MO attribute deletion. If an MO is deleted, any subtree to the deleted MO is also deleted.

7.3.2.1 Delete MO Instances

The `<edit-config>` operation triggers an MO instance deletion request if the `xc:operation="delete"` attribute is present in the class element of an MO instance.

An example on `NtpServer` deletion request is shown in Example 67.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement>
            <managedElementId>NodeName</managedElementId>
            <SystemFunctions>
            <systemFunctionsId>1</systemFunctionsId>
               <SysM>
                  <sysMId>1</sysMId>
                  <NtpServer xc:operation="delete">
                     <ntpServerId>1</ntpServerId>
                  </NtpServer>
               </SysM>
            </SystemFunctions>
         </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>
```

Example 67 Delete MO Operation Request

7.3.2.2 Delete MO Attribute Value

The `<edit-config>` operation triggers an attribute value deletion request if the `xc:operation="delete"` attribute is present in the XML element of an attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement>
            <managedElementId>NodeName</managedElementId>
            <userLabel xc:operation="delete"/>
         </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>
```

Example 68 Delete MO Attribute Operation Request

7.3.2.3 Delete Sequence Element

If a sequence element (multi-value) contains the `xc:operation="delete"` attribute in an `<edit-config>` operation, then the deletion of all the attribute values is triggered.

The deletion of individual sequence elements is not supported unless the following namespace and the position are mentioned:

```
xmlns:erince="urn:com:ericsson:netconf:operation:1.0"
xc:operation="delete" erince:position="all"
```

This functionality is considered as an Ericsson proprietary extension of the NETCONF standard, which only provides one behavior for all combinations of unique/non-unique and ordered/non-ordered multi-values, which is to delete all occurrences of the specified value.

Example 69 show an operational request to delete sequence element.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement>
            <managedElementId>1</managedElementId>
            <exmapleStructAttribute struct="FileDataMv">
             <exmapleStructAttributeId>1</ exmapleStructAttributeId>
             <attrFileData>
               <int64Member xmlns:erince="urn:com:ericsson:netconf:operation:1.0"
               xc:operation="delete" erince:position="all">44</int64Member>
               </attrFileData>
            </exmapleStructAttribute>
         </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>
```

Example 69 Delete Individual Sequence Element Operation Request

7.3.2.4 Delete Struct

If an attribute defined as structure contains the `xc:operation="delete"` attribute in an `<edit-config>` operation, then the deletion of all the attribute values is triggered.

7.3.2.5 Delete Struct Member

If a struct member contains the `xc:operation="delete"` attribute in an `<edit-config>` operation, then the deletion of all the struct member values is triggered. The deletion of individual sequence element in a struct member is also possible, see Section 7.3.2.3 Delete Sequence Element on page 56, with this limitation that it is only possible for a singleton struct. If the struct is not singleton the request is not completed and an error message is returned, as shown in Example 70.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>invalid-value</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">
      Failed to delete attribute value in replace context.
      [dn: ManagedElement=1,exmapleStructAttribute=1, attribute:
      attrFileData]</error-message>
   </rpc-error>
</rpc-reply>
```

Example 70 Error Response for Deleting a Sequence Element in Non-Singleton Struct

7.3.3 <edit–config> Operation with Default Option

If no `operation` attribute is present, the operation is performed according to option `<merge>`, see Section 7.3.4 <edit–config> Operation with <merge> Option on page 58.

7.3.4 <edit–config> Operation with <merge> Option

The configuration data identified by the element containing the `xc:operation="merge"` attribute is merged with the configuration in the running configuration data store.

The merge operation triggers the following:

- MO instance creation request if the MO specified in the `<config>` parameter does not exist in the running configuration.
- MO attribute value assignment request if the MO attribute specified in the `<config>` parameter has no value assigned in the running configuration.
- MO attribute value change request if the MO specified in the `<config>` parameter containing the attribute exists.

The `xc:operation="merge"` option can be omitted as `<merge>` is the default `<edit-config>` operation option.

7.3.4.1 Change Single-Valued Attribute

The operation shown in Example 71 requests to change attribute `administrativeState`.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>NodeName</managedElementId>
        <SystemFunctions>
          <systemFunctionsId>1</systemFunctionsId>
          <SecM xmlns="urn:com:ericsson:ecim:ComSecM">
            <secMId>1</secMId>
            <UserManagement>
              <userManagementId>1</userManagementId>
              <LocalAuthorizationMethod xmlns="urn:com:ericsson:ecim:
              ComLocalAuthorization" xc:operation="merge">
                <localId>1</localId>
                <administrativeState>UNLOCKED</administrativeState>
              </LocalAuthorizationMethod>
            </UserManagement>
          </SecM>
        </SystemFunctions>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>
```

Example 71 Changing administrativeState Attribute

7.3.4.2 Change Sequence Attribute

The operation shown in Example 72 requests to replace the existing sequence elements (`anInt32Attr` is a sequence of int) to the new values. That is, a sequence attribute is treated as one attribute containing several values, thus the attribute is replaced with a new attribute with new values in the merge operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
            <managedElementId>NodeName</managedElementId>
            <NCTest xmlns="urn:com:ericsson:ecim:ncmom1"
            xc:operation="merge">
               <nCTestId>1</nCTestId>
               <anInt32Attr>2</anInt32Attr>
               <anInt32Attr>3</anInt32Attr>
               <anInt32Attr>4</anInt32Attr>
               <anInt32Attr>5</anInt32Attr>
            </NCTest>
         </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>
```

Example 72 Changing Sequence Attribute

Merge of sequence attribute is supported, however, merge of sequence elements is not.

7.3.4.3 Change Sequence of Struct

The attributes defined as a sequence of struct can be changed as shown in Example 73. That is, a sequence attribute (of structs) is treated as one attribute containing several values (structs), thus the attribute is replaced with a new attribute with new values in the merge operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
            <managedElementId>NodeName</managedElementId>
            <SystemFunctions>
               <systemFunctionsId>1</systemFunctionsId>
               <SysM xmlns="urn:com:ericsson:ecim:ComSysM">
                  <sysMId>1</sysMId>
                  <Snmp xmlns="urn:com:ericsson:ecim:ComSnmp"
                  xc:operation="merge">
                     <snmpId>1</snmpId>
                     <agentAddress struct="HostAndPort">
                        <port>162</port>
                        <host>127.0.0.1</host>
                     </agentAddress>
                     <agentAddress struct="HostAndPort">
                        <port>163</port>
                        <host>127.0.0.2</host>
                     </agentAddress>
                  </Snmp>
               </SysM>
            </SystemFunctions>
         </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>
```

Example 73 Changing Sequence of Struct

7.3.4.4 Create MO with Single-Valued Attributes

The operation shown in Example 74 requests to create an MO with single-valued attributes.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement>
            <managedElementId>NodeName</managedElementId>
            <SystemFunctions>
               <systemFunctionsId>1</systemFunctionsId>
               <SysM>
                  <sysMId>1</sysMId>
                  <NtpServer xc:operation="merge">
                     <ntpServerId>1</ntpServerId>
                     <serverAddress>127.0.0.1</serverAddress>
                  </NtpServer>
                  <NtpServer xc:operation="merge">
                     <ntpServerId>2</ntpServerId>
                     <userLabel>userLabelvalue</userLabel>
                     <serverAddress>127.0.0.1</serverAddress>
                  </NtpServer>
               </SysM>
            </SystemFunctions>
         </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>
```

Example 74 Creating MO with Single-Valued Attributes

7.3.4.5 Create MO with Structure Data Type

The operation shown in Example 75 requests to create an MO with a structure data type if the MO does not exist. If the MO exists, the attribute is changed to the new value. That is, the struct is treated as one attribute containing one or more member values that are replaced by the new struct.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement>
            <managedElementId>NodeName</managedElementId>
            <SystemFunctions>
            <systemFunctionsId>1</systemFunctionsId>
               <SysM>
                  <sysMId>1</sysMId>
                  <Snmp>
                     <snmpId>1</snmpId>
                     <agentAddress  struct="HostAndPort">
                        <host>127.0.0.1</host>
                        <port>9980</port>
                     </agentAddress>
                  </Snmp>
               </SysM>
            </SystemFunctions>
         </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>
```

Example 75 Creating MO with Structure Data Type

7.3.4.6 Create MO with Exclusive Structure Data Type

The operation shown inExample 76 requests to create an MO with an exclusive structure member if the MO does not exist. If the MO exists, the attributes are changed to the new values. A structure is exclusive if the structure definition contains the `isExclusive` flag, which means that only one member of the structure can be present in the structure.

The `attrSimpleStruct` attribute is defined as an exclusive structure of `hostId` and `ipAddress` structure members. The example inExample 76 shows a valid operation configuration, as only one structure member is present.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
  <target>
    <running/>
  </target>
  <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ManagedElement>
      <managedElementId>NodeName</managedElementId>
        <NCTest>
          <nCTestId>1</nCTestId>
          <ExclusiveThing xc:operation="create">
            <exclusiveThingId>1</exclusiveThingId>
            <attrSimpleStruct>
              <hostId>testId</hostId>
            </attrSimpleStruct>
          </ExclusiveThing>
        </NCTest>
    </ManagedElement>
  </config>
</edit-config>
</rpc>]]>]]>
```

Example 76 Creating MO with Exclusive Structure Data Type

Example 77 shows an invalid configuration, as two structure members are present.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
  <target>
    <running/>
  </target>
  <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ManagedElement>
      <managedElementId>NodeName</managedElementId>
        <NCTest>
          <nCTestId>1</nCTestId>
          <ExclusiveThing xc:operation="create">
            <exclusiveThingId>1</exclusiveThingId>
            <attrSimpleExclusiveStruct>
              <hostId>testId</hostId>
              <ipAddress>127.0.0.1</ipAddress>
            </attrSimpleExclusiveStruct>
          </ExclusiveThing>
        </NCTest>
    </ManagedElement>
  </config>
</edit-config>
</rpc>]]>]]>
```

Example 77 Violation of Exclusivity

Example 78 shows an error message indicating violation of exclusivity.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>invalid-value</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">
      Violation of exclusivity for attrSimpleExclusiveStruct
      MOC: ExclusiveThing</error-message>
   </rpc-error>
</rpc-reply>
```

Example 78 Error Indication on Violation of Exclusivity

7.3.4.7 Set ECIM Password Attribute

For an `<edit-config>` request, configuring an ECIM password attribute value differs based on the existence of `<cleartext/>` tag as follows:

- If ECIM password type attribute is configured with `cleartext` and with password attribute value in the NETCONF request as shown in Example 79, then password attribute value is encrypted and the encrypted password value is stored.
- If ECIM password type attribute is configured without `cleartext` and with password attribute value in the NETCONF request as shown in Example 80, then password attribute value is stored as it is (that is, what ever value is provided to the password attribute).

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
            <managedElementId>1</managedElementId>
            <SystemFunctions>
               <systemFunctionsId>1</systemFunctionsId>
               <SysM xmlns="urn:com:ericsson:ecim:ComSysM">
                  <sysMId>1</sysMId>
                  <Snmp xmlns="urn:com:ericsson:ecim:ComSnmp">
                     <snmpId>1</snmpId>
                     <SnmpTargetV3>
                        <snmpTargetV3Id>1</snmpTargetV3Id>
                        <privKey struct="EcimPassword">
                           <password>passphrase1</password>
                           <cleartext/>
                        </privKey>
                     </SnmpTargetV3>
                  </Snmp>
               </SysM>
            </SystemFunctions>
         </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>
```

Example 79 Request for Setting an ECIM Password Attribute with <cleartext/> Tag

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
            <managedElementId>1</managedElementId>
            <SystemFunctions>
               <systemFunctionsId>1</systemFunctionsId>
               <SysM xmlns="urn:com:ericsson:ecim:ComSysM">
                  <sysMId>1</sysMId>
                  <Snmp xmlns="urn:com:ericsson:ecim:ComSnmp">
                     <snmpId>1</snmpId>
                     <SnmpTargetV3>
                        <snmpTargetV3Id>1</snmpTargetV3Id>
                        <privKey struct="EcimPassword">
                           <password>passphrase1</password>
                        </privKey>
                     </SnmpTargetV3>
                  </Snmp>
               </SysM>
            </SystemFunctions>
         </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>
```

Example 80 Request for Setting an ECIM Password without <cleartext/> Tag

7.3.5 <edit–config> Operation with <replace> Option

The configuration data identified by the element containing the `xc:operation="replace"` attribute replaces any related configuration in the configuration data store identified by parameter `<target>`. If no such configuration data exists in the configuration data store, it is created. Only the configuration present in parameter `<config>` and not the entire target configuration is affected.

7.3.5.1 Replace MO Instance

The operation shown in Example 81 requests replacing of an MO instance.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement>
            <managedElementId>NodeName</managedElementId>
            <SystemFunctions>
            <systemFunctionsId>1</systemFunctionsId>
               <SysM>
                  <sysMId>1</sysMId>
                  <NtpServer xc:operation="replace">
                     <ntpServerId>1</ntpServerId>
                     <serverAddress>127.0.0.2</serverAddress>
                  </NtpServer>
               </SysM>
            </SystemFunctions>
         </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>
```

Example 81 Replacing with Single-Valued Attribute

7.3.5.2 Replace MO Attribute

Replacing an MO attribute is not supported.

7.3.5.3 Replace Operation with RFC Compliance Latest Mode

The replace operation replaces the indicated MO and any subtree. A replace operation on an MO can be viewed as a `<delete>` operation followed by a `<create>` operation, where the `<delete>` operation deletes any existing subtree.

The operation shown in Example 82 is a request to replace an MO. If the MO exists, the entire subtree is replaced and changed with the new values. If the MO does not exist, then it is created with the new values.

```
<?xml version="1.0" encoding="UTF-8"?>
   <rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <edit-config>
        <target>
          <running/>
       </target>
       <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
          <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
            <managedElementId>1</managedElementId>
              <NCTest>
               <nCTestId>1</nCTestId>
              <XThing xc:operation="replace">
              <xThingId>1</xThingId>
                 <XXThing>
                    <xXThingId>3</xXThingId>
                 </XXThing>
              </XThing>
            </NCTest></ManagedElement>
       </config>
     </edit-config>
</rpc>]]>]]>
```

Example 82 Request for <edit-config> Replace Operation

The replace operation is not allowed on system-created MOs and throws an error as shown in Example 83 and Example 84.

```
<?xml version="1.0" encoding="UTF-8"?>
   <rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <edit-config>
         <target>
          <running/>
         </target>
      <default-operation>replace</default-operation>
        <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
            <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
          <managedElementId>1</managedElementId>
         </ManagedElement>
       </config>
    </edit-config>
</rpc>]]>]]>
```

Example 83 Request for <edit-config> Operation Replacing System-Created MO

```
<?xml version="1.0" encoding="UTF-8"?>
    <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
        <rpc-error>
           <error-type>application</error-type>
           <error-tag>resource-denied</error-tag>
           <error-severity>error</error-severity>
           <error-message xml:lang="en">Cannot Replace "ManagedElement=1". MO class
           ManagedElement, is system created</error-message>
</rpc-reply>
```

Example 84 Error Response for Replacing a System-Created MO

7.3.6 Error with Target Set to Startup

The `<edit-config>` operation loads or changes all or part of a specified configuration in the running datastore. If the startup datastore is set as the target datastore, COM returns the RPC error message, as shown in Example 85.

```
<error-type>protocol</error-type>
<error-tag>operation-not-supported</error-tag>
<error-severity>error</error-severity>
<error-message xml:lang=\"en\">Failed to build operation</error-message>
```

Example 85 Response when Startup Datastore is Used

7.4 <action> Operation

Action execution can be requested by operation `<action>`, see Section 12 on page 107.

7.5 <create-subscription> Operation

This section describes the RFC 5277-defined `<create-subscription>` operation and the COM NETCONF-specific filter extension, see Section 13.5.1 <create-subscription> on page 113.

Description

This operation initiates an event notification subscription that sends asynchronous event notifications to the initiator of the command in the NETCONF session until the subscription ends.

If the session has an active notification subscription, no operation besides `<close-session>` is processed as the RFC 5277-defined interleave capability is not supported.

If the session had heartbeat capability enabled (through the `hello` message), heartbeat notifications are sent to the subscriber at fixed interval (configurable through `licom_netconf_agent.cfg`, default as 180 seconds).

Parameters

The parameters are the following:

- `<stream>` – optional parameter that indicates which stream of events is of interest. If not present, events in the default `NETCONF` stream are sent, and this is the only supported stream. The NETCONF client can request information on the supported event stream, see Section 7.1.4 Request Event Stream Discovery on page 40.

- `<filter>` – optional parameter that indicates which subset of all possible events is of interest. If not present, all events not precluded by other parameters are sent. The format of this parameter is the same as the format of the filter parameter in the NETCONF protocol operations. Attribute `type` of the filter element must be `subtree`, which is the only supported filter type.

  As a deviation from the standard, element `<filter>` can contain the following COM NETCONF proprietary elements, see Section 13.5.1 <create-subscription> on page 113:

  – `<event>`

  – `<filterType>`

  – `<filterValue>`

  Both the standard compliant and the deviations are supported. However, both cannot be present in the same Create Subscription message.

  Note: The COM proprietary filter and the standard filters cannot be mixed in one subscribe message. The tag `event` is an indicator that it is a proprietary filter.

  If there are delete notifications, where the deleted DN, indicated by the NETCONF delete operation, is a subtree root, the notification behavior depends on the producer of the CM events. If COM is configured for internally generating the events, then only one notification for the root of the

deleted MO tree is generated. If COM is configured for receiving the CM events from the Middle Ware (MW) through the SA, then notifications for the other MOs in the subtree can also be produced, depending on how the MW and the SA are implemented.

Since structures and multi-valued attributes are considered atomic values, selection nodes on the content of these types are not allowed. Content match nodes on the content of these types are meaningful and allowed.

Matching of non–key attributes in non–leaf MOs requires the use of managed object SPI. All other matching is done using string matching on MOs and attributes using data received in the Notification Service from the event router.

Use attribute match construct for attributes that are not expected to change frequently.

- `<startTime>` – optional parameter that triggers the replay feature and indicates that the replay is to start at the time specified. If `<startTime>` is not present, this is not a replay subscription. It is not valid to specify `<startTime>` later than the current time. If the specified `<startTime>` is earlier than the log can support, notification `<replayTimeUnsupported>` (see Section 13.6.2 <replayTimeUnsupported> Notification on page 115) is sent and the replay begins with the earliest available notification. This parameter is of type `dateTime` with time zone and is compliant to RFC 3339 – Date and Time on the Internet: Timestamps.

  That is, UTC time, for example, `YYYY-MM-SS'T'hh:mm:ss.+/-hh':'mm` or `YYYY-MM-SS'T'hh:mm:ssZ`, where `Z` denotes zero time zone offset. Fractional seconds are not supported.

  The date and time of the earliest logged notification is available in `replayLogCreationTime` parameter of the event stream can be retrieved, see Section 7.1.4 Request Event Stream Discovery on page 40.

- `<stopTime>` – optional parameter that is used with the optional replay feature to indicate the newest notifications of interest. If `<stopTime>` is not present, the notifications continue until the subscription ends. `<stopTime>` must be later than `<startTime>`. Values of `<stopTime>` in the future are valid. This parameter is of type `dateTime` with time zone and is compliant to RFC 3339 – Date and Time on the Internet: Timestamps.

  Note: The NETCONF session and the contained notification subscription are ended if the session inactivity time-out is reached. For details on the session inactivity timer, see Section 4.3 Session Inactivity Timer on page 12.

The notification service does not support fractional seconds.

`startTime` and `stopTime` must be specified according to format `YYYY-MM-SS'T'hh:mm:ss.<time offset>`, where `<time-offset>` is an offset with format `+/-hh':'mm` or a `Z` denoting an offset of 00:00.

An example of the time format is `2011-11-03T13:54:26+02:00`.

Positive Response

If the NETCONF server can satisfy the request, the server sends an `<ok>` element.

Negative Response

An `<rpc-error>` element is included within the `<rpc-reply>` if the request cannot be completed for any reason. Subscription requests fail if a filter with invalid syntax is provided or if the name of a non-existent stream is provided.

Examples

The operation in Example 86 requests to create a subscription without filter.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
 </create-subscription>
</rpc>]]>]]>
```

Example 86 Creating Subscription without Filter

A subscription with the filter in Example 87 receives notifications upon creation or deletion of any MO starting with DN `ManagedElement=1,SystemFunctions=1,Snmp`, including the creation or deletion of SNMP MOs. Notifications are also sent if any attribute is changed anywhere on any DN.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
     <filter type="subtree">
       <event>
        <filterType>objectCreated</filterType>
        <filterValue>ManagedElement=1,SystemFunctions=1,Snmp.*
        </filterValue>
      </event>
   <event>
    <filterType>objectDeleted</filterType>
       <filterValue>ManagedElement=1,SystemFunctions=1,Snmp.*
       </filterValue>
   </event>
   <event>
    <filterType>attributeChanged</filterType>
    <filterValue>.*</filterValue>
   </event>
  </filter>
 </create-subscription>
</rpc>]]>]]>
```

Example 87 Creating Subscription with Filter

The operation in Example 88 requests to create a subscription with start and stop time.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <filter type="subtree"/>
  <startTime>2012-11-14T22:40:00+01:00</startTime>
  <stopTime>2012-11-14T22:55:00+01:00</stopTime>
 </create-subscription>
</rpc>]]>]]>
```

Example 88 Creating Subscription with Start and Stop Time

The operation in Example 89 requests to create a subscription with content match and MO selection.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <filter type="subtree">
   <ManagedElement>
    <managedElementId>1</managedElementId>
    <TestRootMoc>
    <YThing>
    <rwattr2>2</rwattr2>
    </YThing>
    </TestRootMoc>
   </ManagedElement>
   <ManagedElement>
    <managedElementId>1</managedElementId>
    <TestRootMoc>
    <YThing/>
    </TestRootMoc>
   </ManagedElement>
  </filter>
 </create-subscription>
</rpc>]]>]]>
```

Example 89 Creating Subscription with Content Match and MO Selection (Standard Subtree Filters)

The operation in Example 90 requests to create a subscription with key attribute match and attribute selection.

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <filter type="subtree">
   <ManagedElement>
    <managedElementId>1</managedElementId>
    <TestRootMoc xmlns="urn:com:ericsson:ecim:com_test_cli_1">
    <testRootMocId>1</testRootMocId>
    </TestRootMoc>
   </ManagedElement>
   <ManagedElement>
    <managedElementId>1</managedElementId>
    <TestRootMoc>
    <YThing>
    <rwattr2/>
    </YThing>
    </TestRootMoc>
   </ManagedElement>
  </filter>
 </create-subscription>
</rpc>]]>]]>
```

Example 90 Creating Subscription with Key Attribute Match and Attribute Selection (Standard Subtree Filters)

7.5.1 Create Subscription for Notification Replay

To subscribe to a replay of past event notifications, `startTime` must be specified. Optionally, `stopTime` can also be specified, in which case the subscription ends at the specified time. If `stopTime` is not specified, the subscription continues until the NETCONF session is ended.

If the specified `<startTime>` is earlier than the log can support, the replay begins with `ReplayStartTimeUnsupported` followed by the earliest available notifications, as shown in Example 91.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2012-11-20T16:32:28+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <eventType xmlns=" ">ReplayStartTimeUnsupported</eventType>
   </events>
</notification>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2012-11-19T16:34:18+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <AVC dn="ManagedElement=NodeName">
         <attr name="userLabel">
            <v>XXXXX-27</v>
         </attr>
      </AVC>
   </events>
</notification>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2012-11-20T15:57:23+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <AVC dn="ManagedElement=NodeName">
         <attr name="siteLocation">
            <v>egewrg</v>
         </attr>
      </AVC>
   </events>
</notification>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2012-11-20T15:57:28+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <AVC dn="ManagedElement=NodeName">
         <attr name="siteLocation"/>
      </AVC>
   </events>
</notification>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2012-11-20T16:32:28+01:00</eventTime>
   <replayComplete xmlns="urn:ietf:params:xml:ns:netmod:notification"/>
</notification>
]]>]]>
```

Example 91 Start Time Earlier Than Log Can Support

7.5.2 Terminate Subscription

A subscription ends if any of the followings conditions are met:

- The subscription end time is reached. The NETCONF session is not closed in this case.
- The NETCONF session is closed because of any reason. For details on session close, see Section 4.4 Session End on page 12.

7.6 <close-session> Operation

Operation `<close-session>` requests graceful termination of the current NETCONF session, see Example 92.

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

Example 92 Closing Current Session

7.7 <kill-session> Operation

Operation `<kill-session>` requests the forced termination of the selected NETCONF session, see Example 93.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <kill-session>
      <session-id>24</session-id>
   </kill-session>
</rpc>]]>]]>
```

Example 93 Killing Session 24

If **<session-id>** is equal to the current or a non-existing session identifier, an ‘‘invalid-value’’ error is returned.

If a NETCONF client receives a `<kill-session>` request for an open session, it ends any operations currently in process, rolls back the ongoing transaction, discards all uncommitted changes in the transaction, releases any locks and resources associated with the session, and closes the session.

An `<ok>` message is returned if the request is completed without error, otherwise, an `<rpc-error>` message is returned that indicates reason of the error.

Operation `<kill-session>` also triggers transaction `abort`, see Section 11.4 End on page 104.

7.8 <copy-config> Operation

Description

The `<copy-config>` operation creates or replaces an entire configuration datastore (target) with the content of another complete configuration datastore (source). If the target datastore exists, it is overwritten. Otherwise, a new one is created, if allowed.

Parameters

The operation has the following parameters:

- Target

  It is the name of the configuration datastore to use as the destination of the `<copy-config>` operation.

- Source

  It is the name of the configuration datastore to use as the source of the `<copy-config>` operation, or the `<config>` element containing the complete configuration to copy.

Two datastore combinations are supported:

- Source is `<running>` and target is `<startup>`.

  That is, copy the `<running>` configuration datastore to the `<startup>` configuration datastore.

- Source is `<startup>` and target is `<running>`.

  That is, copy the `<startup>` configuration datastore to the `<running>` configuration datastore.

Positive Response

If the device was able to satisfy the request, an `<rpc-reply>` is sent that includes an `<ok>` element.

Negative Response

If the request cannot be completed for some reason, an `<rpc-reply>` is sent that includes an `<rpc-error>` element.

7.8.1 Error when Target is Same as Source

The <copy-config> operation requests arguments of source datastore and target datastore to execute the operation. If the same datastore is used for both arguments, COM returns the RPC error message, as shown in Example 94.

```
<error-type>protocol</error-type>
<error-tag>invalid-value</error-tag>
<error-severity>error</error-severity>
<error-message xml:lang=\"en\">target is same as source.</error-message>
```

Example 94 Response when Target is Same as Source

7.8.2 Error when User does not Have Proper Access Right

When a user performs the <copy-config> operation, the system validates its access right. If the user does not have access, or no rule is configured for this user, or faulty rule is associated with this user, COM returns the RPC error message, as shown in Example 95.

```
<error-type>protocol</error-type>
<error-tag>access-denied</error-tag>
<error-severity>error</error-severity>
```

Example 95 Response to User without Proper Access Right

7.8.3 Error when Target/Source is Set to Candidate

Candidate datastore is not supported. If either the target or the source argument is set to candidate datastore for the <copy-config> operation, COM returns the RPC error messages, as shown in Example 96 and Example 97.

```
<error-type>protocol</error-type>
<error-tag>unknown-element</error-tag>
<error-severity>error</error-severity>
<error-info><bad-element>candidate</bad-element></error-info>
<error-message xml:lang=\"en\">
candidate is not a valid target configuration.</error-message>
```

Example 96 Response when Target is Set to Candidate Datastore

```
<error-type>protocol</error-type>
<error-tag>unknown-element</error-tag>
<error-severity>error</error-severity>
<error-info><bad-element>candidate</bad-element></error-info>
<error-message xml:lang=\"en\">
candidate is not a valid source configuration.</error-message>
```

Example 97 Response when Source is Set to Candidate Datastore

7.8.4 Error with Empty/Incorrect Namespace

The <copy-config> operation requires the namespace set to urn:ietf:params:xml:ns:netconf:base:1.0. If incorrect or empty namespace is used, COM returns the RPC error messages as shown in Example 98 and Example 99.

```
<error-type>protocol</error-type>
<error-tag>operation-not-supported</error-tag>
<error-severity>error</error-severity>
<error-message xml:lang=\"en\">
Namespace: urn:some:namespace and element: copy-config not supported</error-message>
```

Example 98 Response to Incorrect Namespace: urn:some:namespace

```
<error-type>protocol</error-type>
<error-tag>operation-not-supported</error-tag>
<error-severity>error</error-severity>
<error-message xml:lang=\"en\">
copy-config Tag not supported</error-message>
```

Example 99 Response to Empty Namespace

7.8.5 Error with Incorrect Child Tag

The `<copy-config>` operation requires `<source>` and `<target>` as parameters. If incorrect tag is included as child element in the message, COM returns the RPC error message, as shown in Example 100.

```
<error-type>protocol</error-type>
<error-tag>unknown-element</error-tag>
<error-severity>error</error-severity>
<error-info><bad-element>target1</bad-element></error-info>
```

Example 100 Response to Use Incorrect <target1> Tag as Child Element

7.8.6 Error with Invalid Source Datastore

The `<copy-config>` operation requires a valid source configuration datastore, it could be either running or startup datastore. If an invalid name is attempted, COM returns the RPC error message, as shown in Example 101.

```
<error-type>protocol</error-type>
<error-tag>unknown-element</error-tag>
<error-severity>error</error-severity>
<error-info><bad-element>rrrrrrrr</bad-element></error-info>
<error-message xml:lang=\"en\">
rrrrrrrr is not a valid source configuration.</error-message>
```

Example 101 Response when Invalid Name “rrrrrrrr” Attempted as Source

7.8.7 Error when Operation is Not Supported by Middleware

The `<copy-config>` function is closely depended on middleware’s support. When the middleware does not support the distinct-startup `<copy-config>` operation, COM returns the RPC error message, as shown in Example 102.

```
<error-type>application</error-type>
<error-tag>operation-not-supported</error-tag>
<error-severity>error</error-severity>
<error-message xml:lang=\"en\">
MafOamSpiDataStore is not implemented.</error-message>
```

Example 102 Response when <copy-config> Operation Not Supported by Middleware

7.9 <delete-config> Operation

Description

The <delete-config> operation deletes a configuration datastore.

Note: The <running> configuration datastore cannot be deleted.

Parameters

The operation has following parameter:

- Target

  It is the name of the configuration datastore to be deleted. The supported datastore for this operation is <startup>.

Positive Response

If the device was able to satisfy the request, an <rpc-reply> is sent that includes an <ok> element.

Negative Response

If the request cannot be completed for some reason, an <rpc-reply> is sent that includes an <rpc-error> element.

7.9.1 Error with Deleting Running Configuration

According to RFC6241, the running configuration datastore cannot be deleted. If the running configuration datastore is set as target for the <delete-config> operation, COM returns the RPC error message, as shown in Example 103.

```
<error-type>protocol</error-type>
<error-tag>unknown-element</error-tag>
<error-severity>error</error-severity>
<error-info><bad-element>running</bad-element></error-info>
<error-message xml:lang=\"en\">
Target configuration running is not allowed for delete-config.</error-message>
```

Example 103 Response with Deleting Running Configuration

7.9.2 Error when Using Inexistent Configuration as Target

The <delete-config> operation requires a target datastore to execute the operation. The only supported target is the startup configuration datastore. When inexistent configuration is attempted, COM returns similar RPC error message, as shown in Example 104.

```
<error-type>protocol</error-type>
<error-tag>unknown-element</error-tag>
<error-severity>error</error-severity>
<error-info><bad-element>junkconfig</bad-element></error-info>
<error-message xml:lang=\"en\">
junkconfig is not a valid target configuration.</error-message>
```

Example 104 Response when Inexistent "junkconfig" Configuration is Used as Target

7.9.3 Error when User Not Have Proper Access Right

When a user performs the `<delete-config>` operation, the system validates its access right. If the user does not have access, or no rule is configured for this user, or faulty rule is associated with this user, COM returns the error message shown in Example 105.

```
<error-type>protocol</error-type>
<error-tag>access-denied</error-tag>
<error-severity>error</error-severity>
```

Example 105 Response to User without Proper Access Right

7.9.4 Error when Target is Set to Candidate

Candidate datastore is not supported. If the target argument is set to candidate datastore for the `<delete-config>` operation, COM returns the RPC error message, as shown in Example 106.

```
<error-type>protocol</error-type>
<error-tag>unknown-element</error-tag>
<error-severity>error</error-severity>
<error-info><bad-element>candidate</bad-element></error-info>
<error-message xml:lang=\"en\">
candidate is not a valid target configuration.</error-message>
```

Example 106 Response when Target is Set to Candidate Datastore

7.9.5 Error with Incorrect Tag as Child

The `<delete-config>` operation only requires `<target>` as parameter. If incorrect tag is included as child in the message, COM returns the RPC error messages, as shown in Example 107 and Example 108.

```
<error-type>protocol</error-type>
<error-tag>invalid-value</error-tag>
<error-severity>error</error-severity>
<error-info><bad-element>source</bad-element></error-info>
<error-message xml:lang=\"en\">source in delete-config.</error-message>
```

Example 107 Response to Using <source> Tag as Child

```
<error-type>protocol</error-type>
<error-tag>unknown-element</error-tag>
<error-severity>error</error-severity>
<error-info><bad-element>junkchild</bad-element></error-info>
```

Example 108 Response to Using Incorrect <junkchild> Tag as Child

7.9.6 Error with Adding More than One Child under <target> Tag

The `<delete-config>` operation supports to delete one configuration datastore at a time. If more than one child is added under `<target>` tag, COM returns the RPC error message, as shown in Example 109.

```
<error-type>protocol</error-type>
<error-tag>unknown-element</error-tag>
<error-severity>error</error-severity>
<error-info><bad-element>startup</bad-element></error-info>
<error-message xml:lang=\"en\">
Only one configuration is allowed in target.</error-message>
```

Example 109 Response to Having More than One Child in <target>

7.9.7 Error when Operation is Not Supported by Middleware

The `<delete-config>` function is closely depended on middleware's support. When the middleware does not support the distinct-startup `<delete-config>` operation, COM returns the RPC error message, as shown in Example 110.

```
<error-type>application</error-type>
<error-tag>operation-not-supported</error-tag>
<error-severity>error</error-severity>
<error-message xml:lang=\"en\">
MafOamSpiDataStore is not implemented.</error-message>
```

Example 110 Response when <delete-config> Operation Not Supported by Middleware

8 NETCONF Notifications

The standard-defined <replayComplete> and <notificationComplete> notifications cannot be filtered out and are always sent to the client that has a subscription.

The <notification> elements always contain one <eventTime> element, as defined by the standard, that represents the time of the event with the time zone represented in UTC local time in a format compliant to RFC 3339 – Date and Time on the Internet: Timestampsand ISO 8601, that is, YYYY-MM-SS'T'hh:mm:ss.*+/-hh':'mm*. Fractional seconds are not supported.

Examples date and time that has Europe/Budapest as time zone: 2015-07-02T09:30:00+01:00 during Winter, according to Central European Time (CET) and 2015-07-02T09:30:00+02:00 during Summer, according to Central European Summer Time (CEST).

Note: NETCONF notifications display eventTime, replayLogCreationTime, replayLogAgedTime in local time format appended with its offset from UTC only. For example: 2015-07-01T09:30:00+08:00.

8.1 <replayComplete> Notification

Notification <replayComplete>, see Example 111, is sent after the last notifications requested by notification replay, as defined in RFC 5277. For details on notification replay, see Section 7.5.1 Create Subscription for Notification Replay on page 71.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2012-11-14T21:51:29+01:00</eventTime>
   <replayComplete xmlns="urn:ietf:params:xml:ns:netmod:notification"/>
</notification>
]]>]]>
```

Example 111 <replayComplete>

8.2 <notificationComplete> Notification

Notification <notificationComplete>, see Example 112, is sent when stopTime is reached if <stopTime> has been specified, as defined in RFC 5277.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2012-11-14T21:54:59+01:00</eventTime>
   <notificationComplete
   xmlns="urn:ietf:params:xml:ns:netmod:notification"/>
</notification>
]]>]]>
```

Example 112 <notificationComplete>

8.3 Non-Standard Notifications

Non-standard notifications are supported, see Section 13.6 New Notifications on page 114.

9 Validation

After the operation request is received, the NETCONF server automatically performs a validity check to ensure that the configuration entered in the operation, the running configuration, and the configuration already entered in the transaction constitute a configuration that complies to the validation rules specified as model properties.

Validation rules defined in `Schematron` and MO cardinality rules are checked by the NETCONF server automatically at transaction commit.

9.1 MO Instance Creation

The configuration is considered as invalid and the operation request is rejected in the following cases:

- The MO instance exists in the configuration data store. If the MO instance exists, an `<rpc-error>` element is returned with an `<error-tag>` value of `data-exists`.

The Example 113 and Example 114 shows an error message of MO instance creation.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>data-exists</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">NtpServer=1 already exist
      </error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 113 Error Message

- The MO is defined as `systemCreated`.
- The number of MO instances after creation is higher than the maximal or lower than the minimal cardinality of the MO defined in the model.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>resource-denied</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">
      MOC: TwoMvStructMv, Cardinality 3 is above [0-2]</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 114 Error Message

9.1.1 Omitting Attributes

The behavior of omitting attributes is described as follows:

- If a mandatory attribute is omitted, the validation of the create request fails.
- An attribute is considered mandatory if it does not have a default value, it cannot have 0 number of values and it is `non-readOnly` (`isReadOnly` set to false).

  (An attribute can have 0 number of values, if in the model, it has the `isNillable` property or it is a sequence with 0 `minLength`.)

  Note: The `<mandatory/>` tags in the model are disregarded.
- If an attribute with default value is missing, the MO instance is created with the default value of the attribute. (For a description of default values for structs, see Section 9.6.2 Struct Attribute with Default Values on page 89.)

  If the default value of the attribute is invalid, the suitable validation error is returned.
- If an attribute has 0 number of values and does not have a default value, it is added with 0 number of values if unspecified.

9.2 MO Instance Deletion

The configuration is considered as invalid and the operation request is rejected in the following cases:

- The MO instance does not exist in the configuration data store.
- The MO is defined as `systemCreated`.
- The number of MO instances after creation is lower than the minimal cardinality of the MO defined in the model.
- The SA or MW rejects the `<delete>` operation because of violations of model rules for MOs in a subtree to the deleted MO.

The Example 115 shows an error message of MO instance deletion.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>resource-denied</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">
      MOC: TwoMvStructMv, Cardinality 1 is below [2-3]</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 115 Error Message

9.3 Change Single-Valued Attributes

The configuration is considered as invalid and the operation request is rejected in the following cases:

- If the MO attribute is defined as `mandatory`, without default values and the MO attribute is specified in the MO creation operation request.
- If the MO attribute or struct member is defined as `string`, and the `string length` is not in the allowed range that is specified in the model as `min` and `max` in `lengthRange`.
- If the MO attribute or struct member is defined as `derivedDataType string`, and the string does not comply to the regular expression specified in the model in the `validationRules` element or the deprecated `validValues` element.

  Note: The format attribute of the rule element, that is, a subelement to the `validationRules` element, must in this case be "posix_ere". Only one rule element of this type is allowed in the `validationRules` element.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>unknown-attribute</error-tag>
      <error-severity>error</error-severity>
      <error-info>
         <bad-attribute>anInt32AttrDerived</bad-attribute>
         <bad-element>StructDerived</bad-element>
      </error-info>
      <error-message xml:lang="en">Failed to add attribute</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 116 Error Message

- If the MO attribute is defined as `integer`, and the integer value is not in the allowed range that is specified in the model as `range/min` and `range/max`.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>unknown-attribute</error-tag>
      <error-severity>error</error-severity>
      <error-info>
         <bad-attribute>anInt32AttrDerived</bad-attribute>
         <bad-element>StructDerived</bad-element>
      </error-info>
      <error-message xml:lang="en">Failed to add attribute</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 117 Error Message

- If the MO attribute is defined as `double`, and the double value is not in the allowed range that is specified in the model as `range/min` and `range/max`.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>unknown-attribute</error-tag>
      <error-severity>error</error-severity>
      <error-info>
         <bad-attribute>rwdouble1Derived </bad-attribute>
         <bad-element> XThingMv </bad-element>
      </error-info>
      <error-message xml:lang="en">Value 1.7977e+308 is out of range for attribute rwdouble1Derived expected
values from Value range exception</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 118 Error Message

- If the MO attribute is defined as `double`, and the double value has resolution specified in the model, it needs to have subrange specified in the model as `range/min` and `range/max` and the value is not valid if it does not fit this formula: `min + resolution * n <= max`

```
<?xml version="1.0" encoding="UTF-8"?>3
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>unknown-attribute</error-tag>
      <error-severity>error</error-severity>
      <error-info>
         <bad-attribute> rwdouble1Derived</bad-attribute>
         <bad-element>XThingMv</bad-element>
      </error-info>
      <error-message xml:lang="en"> Value 40.3 for attribute rwdouble1Derived is not of expected resolution
from [10.5,50.5],[-2.1,2.1] using resolution 0.5</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 119 Error Message

- If the MO attribute is defined as `double`, and the double value has precision larger than 15 digits (this limitation is added to avoid the rounding errors for double datatype).

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>unknown-attribute</error-tag>
      <error-severity>error</error-severity>
      <error-info>
         <bad-attribute> rwdouble1Derived</bad-attribute>
         <bad-element>XThingMv</bad-element>
      </error-info>
      <error-message xml:lang="en"> Value 1.52220000255544846465 has invalid precision for attribute
rwdouble1Derived</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 120 Error Message

- If the MO attribute is defined as `double`, and the double value is one of the following:
  - `positiveZero`
  - `negativeZero`
  - `positiveInfinity`
  - `negativeInfinity`
  - `notANumber`

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>unknown-attribute</error-tag>
      <error-severity>error</error-severity>
      <error-info>
         <bad-attribute>rwdouble1Derived </bad-attribute>
         <bad-element> XThingMv </bad-element>
      </error-info>
      <error-message xml:lang="en">Failed to add attribute</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 121 Error Message

9.4 Delete MO Attribute Values

The configuration is considered as invalid and the operation request is rejected in the following cases:

- The MO attribute does not exist in the configuration data store.
- The MO attribute is not defined in the model as `isNillable` or the attribute is defined as `mandatory`, that is, its minimal cardinality is not zero.
- The MO attribute is defined as `readOnly` or `restricted`.

9.5 Change Sequence Attributes

The configuration is considered as invalid and the operation request is rejected in the following cases:

- If an attribute is defined as sequence, and the number of sequence elements is out of the range defined by the model as `minLength` and `maxLength`.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>invalid-value</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">
      Multiplicity violation, attribute name: attrFileData,anInt32Attr
      MOC: SimpleStructMv</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 122 Error Message

- If an attribute is defined as sequence with unique sequence elements (**nonUnique** is not present), and multiple sequence elements contain the same value.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>invalid-value</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">
      Violation of uniqueness for attrFileData
      MOC: SimpleStructMv</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 123 Error Message

9.6 Change Struct Attributes

The configuration is considered as invalid and the operation request is rejected in the following cases:

- If structure member data types and the number of structure member values are not according to the model definition.
- If a structure is defined as exclusive (`isExclusive`), and the structure contains multiple structure members. See Example 124.

Merge of individual struct members on existing singleton structs is supported by ebase capability `urn:com:ericsson:ebase:1.2.0` and later. Lower ebase capabilities require the complete struct with all required values to be present (except for nillable members) in the operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>invalid-value</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">
      Violation of exclusivity for attrSimpleExclusiveStruct
      MOC: ExclusiveThing</error-message>
   </rpc-error>
</rpc-reply>
```

Example 124 Error Message

9.6.1 Omitting Members in Structs

If a struct has an unspecified member, the following apply:

- Operation is create (or) replace:
  - If the struct member is mandatory and does not have a default value, then the configuration is rejected.
  - If it has a default value, the member is added to the configuration with the default value and if there is no default value then the member is not added to the configuration.
- Operation is merge, which does not result in MO creation,
  - For singleton struct attributes, the omitted struct members are fetched from the database, if ebase capability `urn:com:ericsson:ebase:1.2.0` and later used. See Example 125.
  - For singleton struct attributes, for lower ebase capabilities, the complete struct with all the struct members (except for nillable) are to be present in the operation, otherwise the operation leads to error. See Example 126.
  - For sequence of struct attributes, if there are any omitted struct members (except for nillable), the operation leads to error for all the capabilities. See Example 127.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
          <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
          <managedElementId>1</managedElementId>
          <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
          <nCTestId>1</nCTestId>
          <SimpleStruct>
          <simpleStructId>1</simpleStructId>
            <attrFileData struct="FileData">
               <aBoolAttrNoDefault>true</aBoolAttrNoDefault>
               <anInt32Attr>10</anInt32Attr>
               <aBoolAttr>false</aBoolAttr>
            </attrFileData>
           </SimpleStruct>
           </NCTest>
           </ManagedElement>
       </config>
   </edit-config>
</rpc>
]]>]]>

  <?xml version="1.0" encoding="UTF-8"?>
  <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
     <ok/>
  </rpc-reply>
  ]]>]]>
```

Example 125 Merge Operation Successful with ebase 1.2.0 (Singleton struct)

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
        <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
         <managedElementId>1</managedElementId>
          <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
           <nCTestId>1</nCTestId>
           <SimpleStruct>
             <simpleStructId>1</simpleStructId>
               <attrFileData struct="FileData">
                  <aBoolAttrNoDefault>true</aBoolAttrNoDefault>
                  <anInt32Attr>10</anInt32Attr>
                  <aBoolAttr>false</aBoolAttr>
               </attrFileData>
           </SimpleStruct>
           </NCTest>
          </ManagedElement>
         </config>
       </edit-config>
 </rpc>
 ]]>]]>

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
  <rpc-error>
   <error-type>application</error-type>
     <error-tag>invalid-value</error-tag>
     <error-severity>error</error-severity>
     <error-message xml:lang="en">Multiplicity violation, attribute name: attrFileData,anInt64Attr
      MOC: SimpleStructMv
     </error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 126 Merge Operation Failed with ebase 1.1.0 (Singleton struct)

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
  <edit-config>
    <target>
      <running/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>1</managedElementId>
          <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
            <nCTestId>1</nCTestId>
          <SimpleStruct>
             <simpleStructId>1</simpleStructId>
             <attrFileData2 struct="FileData">
               <aBoolAttrNoDefault>true</aBoolAttrNoDefault>
               <anInt32Attr>10</anInt32Attr>
               <anInt64Attr>50</anInt64Attr>
               <aBoolAttr>false</aBoolAttr>
               <aStringAttr>Trollmor</aStringAttr>
             </attrFileData2>
             <attrFileData2 struct="FileData">
             <aBoolAttrNoDefault>true</aBoolAttrNoDefault>
             <aBoolAttr>false</aBoolAttr>
             </attrFileData2>
         </SimpleStruct>
        </NCTest>
      </ManagedElement>
     </config>
   </edit-config>
</rpc>
]]>]]>

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-message xml:lang="en">Multiplicity violation, attribute name: attrFileData2,anInt32Attr
     MOC: SimpleStructMv
    </error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

Example 127 Merge Operation Failed with Any ebase (sequence of structs)

9.6.2 Struct Attribute with Default Values

The struct attribute with default values is as follows:

- The default value of a struct attribute is derived from the default values of its struct members. If any of the struct members are mandatory, then the struct attribute does not have a default value either.
- The default value of a struct has the minimum number of values of the struct attribute and every struct value is constructed with the members having their default values or 0 number of values for those members that do not have a default.

9.7 System-Created Property

The `isSystemCreated` property on `EcimMoClass` is deprecated in `MetaModel` Rev C.

This property is replaced by canCreate/canDelete properties on EcimContainment and EcimContribution.

Properties canCreate, canDelete for a EcimContainment/Contribution are supported from DX ET12.0 only.

The Table 5 shows the properties and their respective tags.

Table 5 Properties and Their Respective Tags

Property (While Modelling)	Tag (mp.xml File)
canCreate = false	*<notCreatable/>*
canDelete = false	*<notDeleteable/>*
isSystemCreated = true	*<systemCreated/>*

When isSystemCreated, canCreate, and canDelete properties are true, isSystemCreated property has the precedence and in rest all cases, CanCreate/canDelete properties have the precedence.

Properties canCreate and canDelete are used in the following NETCONF operations:

- Create

  The canCreate property is used to check whether a MOC is creatable, see Section 9.7.1 Validation for MOC Creation on page 91.

- Delete

  The canDelete property is used while deleting a MOC, see Section 9.7.2 Validation for MOC Deletion on page 93.

- Replace (RFC)

  Operation replaces as per RFC involves both operation delete followed by operation create. So, both the properties canDelete and canCreate are verified while replacing a MOC, see Section 9.7.3 Validation for MOC Replace with RFC Compliance Latest Mode on page 94.

- Get-Config

  Property canCreate (or) SystemCreated (in legacy models) is required for deciding on configuration data. For definition of configuration and state data, see Section 6.3 Configuration and State Data on page 22.

Example 128 shows a Test_Mp.xml, which contains <notCreatable/> and <notDeleteable/> tags in the relationship between MOC and its parent.

```
<relationship name="TestRootMoc_to_SimpleMoc">
   <containment>
      <parent>
         <hasClass name="TestRootMoc">
         </hasClass>
      </parent>
      <child>
         <hasClass name="SimpleMoc">
         </hasClass>
         <cardinality>
            <min>0</min>
          </cardinality>
       </child>
       <notCreatable/>
       <notDeleteable/>
   </containment>
</relationship>
```

Example 128 Relationship between MOC and Its Parent

9.7.1 Validation for MOC Creation

The configuration is considered as invalid and the create operation request is rejected, if a MOC complies to any of the following:

- Has `<systemCreated/>` tag and there are no `<notCreatable/>` and `<notDeleteable/>` tags in the relationship between the MOC and its parent.
- Does not have `<systemCreated/>` tag, but the relationship with its parent has `<notCreatable/>` tag.
- Error messages are thrown based on the existence of `<systemCreated/>`.

Error messages are the following:

- If a MOC has `<systemCreated/>` tag, then ‘‘Old’’ error-message is thrown. Old: cannot instantiate ‘‘<mocName>=<instance>’’. MO class `<mocName>` is system-created.
- If a MOC does not have `<systemCreated/>` tag, then ‘‘New’’ error-message is thrown. New: cannot create ‘‘<mocName>=<instance>’’. MO class `<mocName>` is not creatable.

Table 6 shows the error messages for invalid create operation.

Table 6 Error Messages for Invalid Create Operation

S.No.	isSystemCreated	canCreate	canDelete	Error Message (New/Old)
1	TRUE	TRUE	TRUE	Old
2	TRUE	FALSE	FALSE	Old
3	TRUE	FALSE	TRUE	Old
4	FALSE	FALSE	FALSE	New
5	FALSE	FALSE	TRUE	New

The <edit-config> operation, as shown in Example 129, is a request to create MO SimpleMOC. The error messages, that are thrown when the previous request is executed, are described in Example 130 and Example 131.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement>
            <managedElementId>1</managedElementId>
               <TestRootMoc>
                  <testRootMocId>1 </testRootMocId>
                     <SimpleMoc xc:operation="create">
                        <simpleMocId>2</simpleMocId>
                     </SimpleMoc>
               </TestRootMoc>
         </ManagedElement>
     </config>
  </edit-config>
</rpc>
```

Example 129 Request for Creating a MOC

If SimpleMoc has <systemCreated/> tag, an old error message is thrown as shown in Example 130.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
   <message-id="1">
      <rpc-error>
         <error-type>application</error-type>
         <error-tag>resource-denied</error-tag>
         <error-severity>error </error-severity>
       <error-message xml:lang="en">Can not instantiate "SimpleMoc=2". MO
                class SimpleMoc, is system created</error-message>
      </rpc-error>
   </rpc-reply>
```

Example 130 Error Response for Create Operation When MOC Has <systemCreated/> Tag

If SimpleMoc does not have <systemCreated/> tag, then new error message is thrown as shown in Example 131.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
   <message-id="1">
      <rpc-error>
         <error-type>application</error-type>
         <error-tag>resource-denied</error-tag>
         <error-severity>error </error-severity>
         <error-message xml:lang="en"> Can not create "SimpleMoc=2".
          MO class SimpleMoc, is not creatable
         </error-message>
      </rpc-error>
   </rpc-reply>
```

Example 131 Error Response for Create Operation When MOC Does Not Have <systemCreated/> but <notCreatable/> Tag

9.7.2 Validation for MOC Deletion

The configuration is considered as invalid and the delete operation request is rejected, if a MOC applies to any of the following:

- Has `<systemCreated/>` tag and there are no `<notCreatable/>` and `<notDeletable/>` tags in the relationship between the MOC and its parent.
- Does not have `<systemCreated/>` tag, but the relationship between the MOC and its parent has `<notDeleteable/>` tag.

Error messages are thrown based on the existence of `<systemCreated/>`.

Error messages are the following:

- If the MOC has `<systemCreated/>` tag, then “Old” error-message is thrown. Old: cannot delete “<mocName>=<instance>”. MO class `<mocName>` is system-created.
- If the MOC does not have `<systemCreated/>` tag, then “New” error-message is thrown. New: cannot delete “<mocName>=<instance>”. MO class `<mocName>` is not deletable.

Table 7 shows the error messages for invalid create operation.

Table 7 Error Messages for Invalid Delete Operation

S.No.	isSystemCreated	canCreate	canDelete	Error Message (New/Old)
1	TRUE	TRUE	TRUE	Old
2	TRUE	TRUE	FALSE	Old
3	TRUE	FALSE	FALSE	Old
4	FALSE	TRUE	FALSE	New
5	FALSE	FALSE	FALSE	New

The `<edit-config>` operation, as shown in Example 132, is a request to delete MO `SimpleMOC`. The error messages, that are thrown when the previous request is executed, are described in Example 133 and Example 134.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement>
            <managedElementId>1</managedElementId>
               <TestRootMoc>
                  <testRootMocId>1</testRootMocId>
                     <SimpleMoc xc:operation="delete">
                        <simpleMocId>1</simpleMocId>
                     </SimpleMoc>
                </TestRootMoc>
          </ManagedElement>
       </config>
    </edit-config>
 </rpc>
```

Example 132 Request for Deleting a MOC

If `SimpleMoc` has `<systemCreated/>` tag, an old error message is thrown, as shown in Example 133.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>resource-denied</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Can not delete "SimpleMoc=1". MO class
      SimpleMoc, is system created</error-message>
   </rpc-error>
</rpc-reply>
```

Example 133 Error Response for Delete Operation When MOC Has <systemCreated/> Tag

If `SimpleMoc` does not have `<systemCreated/>` tag, then new error message is thrown, as shown in Example 134.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>resource-denied</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Can not delete "SimpleMoc=1".
      MO class SimpleMoc, is not deletable</error-message>
   </rpc-error>
</rpc-reply>
```

Example 134 Error Response for Delete Operation When MOC Does Not Have <systemCreated/> but <notDeleteable/> Tag

9.7.3 Validation for MOC Replace with RFC Compliance Latest Mode

The configuration is considered as invalid and the replace operation request in RFC Compliance Latest Mode is rejected, if a MOC applies to any of the following:

- Has `<systemCreated/>` tag and there are no `<notCreatable/>` and `<notDeletable/>` tags in the relationship between the MOC and its parent.

- There is no <systemCreated/> tag for the MOC, but the relationship between the MOC and its parent has <notDeleteable/> and <notCreatable/> tag.

Error messages are thrown based on the existence of <systemCreated/>.

Error messages are the following:

- If the MOC has <systemCreated/> tag, the "Old" error message is thrown. Old: cannot replace "<mocName>=<instance>". MO class <mocName> is system created.
- If the MOC does not have <systemCreated/> tag, but contains <notDeleteable/> tag, the "Deletable" error message is thrown. Deletable: cannot replace "<mocName>=<instance>". MO class <mocName> is not deletable.
- If the MOC does not have <systemCreated/> tag, but contains <notCreatable/> tag, then the "Creatable" error message is thrown. Creatable: cannot replace "<mocName>=<instance>". MO class <mocName>, is not creatable.

Table 8 shows the error messages for invalid create operation.

Table 8 Error Messages for Invalid Replace Operation in RFC Compliance Latest Mode

S.NO.	isSystemCreated	canCreate	canDelete	Error Message (New/Old)
1	TRUE	TRUE	TRUE	Old
2	TRUE	TRUE	FALSE	Old
3	TRUE	FALSE	TRUE	Old
4	TRUE	FALSE	FALSE	Old
5	FALSE	TRUE	FALSE	Deletable
6	FALSE	FALSE	TRUE	Creatable
7	FALSE	FALSE	FALSE	Deletable

The <edit-config> operation, as shown in Example 135, is a request to replace MO SimpleMOC. The error messages, that are thrown when the previous request is executed, are described in Example 136, Example 137, and Example 138.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ManagedElement>
            <managedElementId>1</managedElementId>
               <TestRootMoc>
                  <testRootMocId>1 </testRootMocId>
                     <SimpleMoc xc:operation="replace">
                        <simpleMocId>1</simpleMocId>
                     </SimpleMoc>
               </TestRootMoc>
         </ManagedElement>
       </config>
    </edit-config>
 </rpc>
```

Example 135 Request for Replacing a MOC

If `SimpleMoc` has `<systemCreated/>` tag, an old error message is thrown, as shown in Example 136.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>resource-denied</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Cannot Replace "SimpleMoc=1".
       MO class SimpleMoc, is system created</error-message>
</rpc-reply>
```

Example 136 Error Response for Replace (RFC Compliance Latest Mode) Operation When MOC Has <systemCreated/> Tag

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>resource-denied</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Cannot Replace "SimpleMoc=1".
       MO class SimpleMoc, is not deletable</error-message>
   </rpc-error>
</rpc-reply>
```

Example 137 Error Response for Replace (RFC Compliance Latest Mode) Operation When MOC Does Not Have <systemCreated/> but <notDeleteable/> Tag

If `SimpleMoc` does not have `<systemCreated/>` tag, but contains only `<notCreatable/>` tag, then error message is thrown, as shown in Example 138.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>resource-denied</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Cannot Replace "SimpleMoc=1".
       MO class SimpleMoc, is not creatable</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 138 Error Response for Replace (RFC) Operation When MOC Does Not Have <systemCreated/> but Only <notCreatable/> Tag

9.8 Passphrase Property

The `isPassphrase` property is introduced for an `EcimDerivedString` in *ECIM MetaModel Specification Rev C* and supported from DX ET12.0 only.

If an `EcimDerivedString` attribute contains the `isPassphrase` property, the string holds a `passphrase` and its value must not be revealed.

Table 9 shows the Passphrase property and its Respective tag.

Table 9 Passphrase Property and its Respective Tag

Property (While Modelling)	**Tag (mp.xml File)**
isPassphrase = true	*<isPassphrase/>*

NETCONF behavior for attributes having this property varies based on the presence of *<enableEncryptedPassphraseInputOutput/>* in the NETCONF component `cfg` file. This is explained in Table 10 and in Table 11

Table 10 shows the NETCONF behavior for `Passphrase` attributes when *<enableEncryptedPassphraseInputOutput/>* is not present in the configuration file.

Table 10 NETCONF Behavior for Passphrase Attributes (in the Absence of <enableEncryptedPassphraseInputOutput/>)

Attribute Type	**Setting (Edit-Config)**	**Retrieving (Get/Get-Config)**	**Notification Display**
KeyAttribute (MOC/Struct)	Property is ignored	Property is ignored and the value is displayed as it is given in the request. (See Example 139, where the key attribute, *<passphraseThingId>* has the `isPassphrase` property set to true)	Property is ignored and the value is displayed as it is given in the request.
Normal attributes (MOC/Struct)	Value is encrypted. Logging is provided with 8*'s irrespective of the length of the attribute value.	8*'s is displayed (See Example 139, where the attribute, `rwatrp`, has the `isPassphrase` property set to true)	8*'s is displayed (See Example 140, where the attribute, `rwatrp`, has the `isPassphrase` property)
Action parameter /Return-value	Property is ignored. Parameter values are sent to the action implementer in clear-text format without encrypting.	Property is ignored. Return value containing `passphrase` property is ignored and the value is printed as it is.	Not Applicable

Table 11 shows the NETCONF behavior for `Passphrase` attributes when *<enableEncryptedPassphraseInputOutput/>* is present in the configuration file.

Table 11 NETCONF Behavior for Passphrase Attributes (in the Presence of <enableEncryptedPassphraseInputOutput/>)

Attribute Type	Setting (Edit-Config)	Retrieving (Get/Get-config)	Notification Display
Key attribute (MOC/Struct)	Property is ignored	Property is ignored	Property is ignored
Normal attributes (MOC/Struct)	Value is encrypted (according to an internal algorithm) before storing in the datastore.	Value is displayed in `encrypted:{value}`, where `{value}` is in encrypted format (according to an internal algorithm).	Value is displayed as `encrypted:{value}`, where `{value}` is in encrypted format (according to an internal algorithm).
Action Parameter /Return-value	Property is ignored	Property is ignored	Not Applicable

Example 139 shows the *<get>* response for attribute containing passphrase property.

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
         <managedElementId>1</managedElementId>
            <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
               <nCTestId>1</nCTestId>
                  <PassphraseThing xmlns="urn:com:ericsson:ecim:Passphrasething">
                      <passphraseThingId>2</passphraseThingId>
                         <rwattrp>********</rwattrp>
                  </PassphraseThing>
            </NCTest>
      </ManagedElement>
   </data>
</rpc-reply>
]]>]]>
```

Example 139 <get> Response for Attribute Containing passphrase Property

Example 140 shows the *<notification>* for `isPassphrase` property attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2015-07-23T12:22:43Z</eventTime>
   <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <AVC dn="ManagedElement=1,NCTest=1,PassphraseThing=2">
       <attr name="rwattrp">
          <v>********</v>
       </attr>
      </AVC>
   </events>
</notification>
]]>]]>
```

Example 140 <notification> for isPassphrase Property Attribute

9.8.1 Delete Passphrase Attributes

If the delete request contains `Passphrase` attribute with value, an error is thrown irrespective of whether the value is present in database.

See Example 141, where `rwattrp` has `isPassphrase` property set.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
        <managedElementId>1</managedElementId>
          <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
            <nCTestId>1</nCTestId>
            <PassphraseThing >
               <passphraseThingId>2</passphraseThingId>
               <rwattrpp2 xmlns:erince="urn:com:ericsson:netconf:operation:1.0" xc:operation="delete"
                erince:position="all"> 44 </rwattrpp2>
            </PassphraseThing>
          </NCTest>
        </ManagedElement>
      </config>
   </edit-config>
</rpc>
]]>]]>

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>operation-not-supported</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Delete operation with specified MO attribute value with
       position "all" is not supported for MO attribute "rwattrpp2".</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 141 Delete Passphrase Attribute with CDATA Given in the Request

9.9 ManagedElement ID Translation

The COM NETCONF agent must accept any value of `ManagedElement` ID in NETCONF requests. It can automatically correct this to the actual/correct ID.

NETCONF responses are unaffected and must contain the correct value. This means if the `networkManagedElementId` is set, the value is retrieved from this attribute, otherwise the value used in the MW or database is returned.

10 NETCONF Visibility Control

The validity of the operations and the returned elements in the reply message depend on the visibility determined by the model and the visibility controller configuration in *Life Cycle Management*. For more information, refer to *Glossary of Terms and Acronyms*. Operations `<get>`, `<get-config>`, `<edit-config>`, and `<action>` are not allowed to address MO, MO attribute, or MO action elements if `NOT_VISIBLE` is assigned to their life cycle specifier tag in the configuration file.

The behavior from different operations and requests on an invisible element is different. For example, an error message is received when trying to delete, create, replace, merge, or get an invisible element in the model by performing an `<edit-config>` or `<get>` operation. This indicates the directly addressed element that is not visible, while an invisible element like an attribute is skipped in both `<get>` and `<get-config>` operations that are performed on a visible MO containing the invisible attribute. Likewise non-visible MOs in subtrees are skipped when traversing a subtree initiated by a `<get>` or `<get-config>` operation.

Note: NETCONF does not differentiate between `ACCESSIBLE` and `VISIBLE`. The entities tagged `ACCESSIBLE` or `VISIBLE` are accessible through NETCONF.

11 Transaction

Operations in the NETCONF interface are performed in an atomic way in transactions.

11.1 Transaction Start

A new transaction automatically starts when a NETCONF session is created, or when a previous transaction in the session has been ended or committed.

11.2 Locking

When an MO is locked, the following is prohibited by other concurrent sessions on the MO:

- Creation/deletion.
- Deletion of its child MOs.
- Modification of its attributes.
- Creation of child MOs.
- Execution of any action which contains this MO.

In the following cases, the MO gets locked:

- If an attribute change is requested, the MO that contains the attribute is locked.
- If an MO creation is requested, the MO and its parent MO are locked.
- If an MO deletion is requested, the MO, its parent MO, and its children MO are locked.
- If an action execution is requested, the MO that contains the action is locked.

The locks are released if the transaction is committed or ended, or if the result of action execution is received. The locking behavior, as described in the previous bullets, is not implemented by NETCONF, but by the one implementing the actual MO operated on. That is normally the SA/MW, or in some cases other entities providing implementation of a fragment of the model.

11.3 Transaction Commit

A transaction represents changes done through <edit-config> messages. A transaction commit triggers validation of the changes entered in the transaction and applies changes to the node if the validation succeeds.

Depending on which capabilities that were exchanged in the <hello> messages at the beginning of the NETCONF session, and the configuration of the ports used, transaction commit can take place either at the <close-session> message or at each <edit-config> message. For more information, see Section 5.1.2 on page 16.

As a response, message <ok> indicates that the transaction is completed without error. Message <rpc-error> indicates that error occurred during validation or applying the changes.

- A failed commit operation is shown in Example 142.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>operation-failed</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Transaction commit failed</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 142 Commit Operation Failed

- A commit operation can be failed if a mandatory child MO is not created for the parent MO in the same transaction, as shown in Example 143.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="999">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>operation-failed</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Parent DN:ManagedElement=1,CMinTop=1,
       child MOC: RcMin1, Cardinality 0 is below [1-10]</error-message>
   </rpc-error>
</rpc-reply>
```

Example 143 Commit Operation Failed Because of Missing Mandatory Child MO

Note: If a parent MO has a mandatory child MO that has model property system-created, then the mandatory child MO is not validated for existence before commit. In this scenario, commit succeed.

11.4 End

A transaction is ended if the NETCONF session is closed without operation <close-session>, that is, a session is closed by the following events:

- Inactivity time-out
- Operation <kill-session>

- OAM network disruption between the client and the server

11.5 Transaction Time-Out

To protect system resources, transaction lifetime is limited by the `transaction inactivity timeout` mechanisms. The transaction times out if the server detects no activity in the transaction during period of 3600 seconds. The timer is not started until a potentially locking MO operation is performed (`create` MO, `delete` MO, `set` attribute, or action). Activity means any message exchange, like `<get>`, `<get-config>`, `<edit-config>`, `<action>` operation request or operation results. At transaction inactivity time-out, the following events are triggered:

- Transaction end that results in removal of transaction content and MO locks.
- State of the transaction is changed from active to timed out.

Operations entered in a timed out transaction is replied with a transaction time-out error message.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>data-missing</error-tag>
        <error-severity>error</error-severity>
        <error-message xml:lang="en">
        MO: ManagedElement=1,SystemFunctions=1
        is not available (no instance).</error-message>
    </rpc-error>
</rpc-reply>
]]>]]>
```

Example 144 Sample Error Message on Transaction Time-Out to <get> Request Execution

If the session is active after the error message is received when the transaction timeout occurring scenarios, close the transaction and the session, see Section 7.6 <close-session> Operation on page 72.

If the session inactivity timer occurs before transaction time-out, the session is closed and the transaction is immediately ended.

12 <action> Capability

This section describes the COM NETCONF-specific `:action` capability. The description is based on the standard capability template defined in RFC 6241 – Network Configuration Protocol (NETCONF).

12.1 Overview

This capability introduces one new `<rpc>` method that is used to start actions defined in the information model.

12.2 Dependencies

None.

12.3 Capability Identifier

The action capability is identified by the following capability string:

`urn:ericsson:netconf:capability:action:1.0`

12.4 New Operations

This section describes the `<action>` operation.

12.4.1 <action>

Description

NETCONF client sends an action request in an `<rpc>` message and the request is replied by a single `<rpc-reply>` message containing the same `message-id` as sent by the client in the request message.

Operation Request and Parameters

The action request message consists of the following XML elements:

- An `<rpc>` element that contains the standard defined `message-id` and `xmlns` XML attributes.
- The `<rpc>` element contains one`<action>` element with the `xmlns="urn:com:ericsson:ecim:1.0"` name space definition.

- The <action> element contains one <data> element.
- The <data> element contains the DN of the MO the action is started on. The class elements in the DN can contain the xmlns="urn:com:ericsson:ecim:<model name>" namespace definitions.
- The last DN element contains one action element. The name of the action element is identical to the action name.
- The action element contains zero or more action parameter elements according to the action definition in the information model. The name of the action parameter element is identical to the action parameter name. Action parameters can be of simple or complex type that is represented according to the notation of the generic COM NETCONF data model. The action parameters can be sent in any sequence order and an action parameter can be omitted if it has default value defined.

Positive Response

If the action return value is defined as void, a standard <ok/> message is returned to indicate that the operation has completed without error.

If the action return value is defined as not void, then the return values are sent encapsulated in a <returnValue> element in the <rpc-reply> message. Action return values can be of simple or complex type that is represented according to the notation of the generic COM NETCONF data model.

The action return value message consists of the following XML elements:

- An <rpc> element that contains the standard defined message-id and xmlns XML attributes.
- The <rpc> element contains one <data> element.
- The <data> element contains the DN of the MO the action is started on. The class elements in the DN contain the xmlns="urn:com:ericsson:ecim:<model name>" namespace definitions.
- The last DN element contains one action element. The name of the action element is identical to the action name.
- The action element contains one <returnValue> element that contains either a single return value or further return value elements according to the action definition in the information model. Action return values are represented according to the notation of the generic COM NETCONF data model.

Depending on the semantics, the success of an action request can mean that the action execution is completed or started that results in asynchronous or synchronous action execution.

The start of the actual action execution can be bound to various conditions depending on the action type, for example, as follows:

- Immediate Start – The action execution starts immediately after the request is entered.
- Start by Commit – The action execution starts after the request is entered and the related transaction is committed.
- Confirmed Start – To start the action execution, additional confirmation is needed.
- Start with Timer – The action execution starts after a predefined time-out. This can be combined with a confirmed start.
- Any action-specific preconditions, for example, internal states and parallel activities.

Negative Response

If the action request cannot be completed for any reason, a standard `<rpc-error>` included in a `<rpc-reply>` message is returned. Action exceptions are not supported.

The `<rpc-error>` contains the following elements:

- `error-type` that is always `application`
- `error-tag` that contains one of the standard defined values
- `error-severity` that is always `error`
- `error-info` that identifies the invalid message element
- `error-message` that provides textual description on the error

Examples

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <action xmlns="urn:com:ericsson:ecim:1.0">
    <data>
        <ManagedElement>
            <managedElementId>NodeName</managedElementId>
            <TestRootMoc>
                <testRootMocId>1</testRootMocId>
                <CallableThing>
                    <callableThingId>1</callableThingId>
                    <addNumbers>
                      <num1>1</num1>
                      <num2>2</num2>
                    </addNumbers>
                </CallableThing>
            </TestRootMoc>
        </ManagedElement>
    </data>
   </action>
</rpc>]]>]]>
```

Example 145 Action Request

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="1">
   <data>
      <ManagedElement>
         <managedElementId>NodeName</managedElementId>
         <NCTest>
            <nCTestId>1</nCTestId>
            <ActionMoc>
               <actionMocId>1</actionMocId>
               <actionName>
                  <returnValue>3</returnValue>
                  <returnValue>4</returnValue>
                  <returnValue>5</returnValue>
               </actionName>
            </ActionMoc>
         </NCTest>
      </ManagedElement>
   </data>
</rpc-reply>
]]>]]>
```

Example 146 Multiple Return Values

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="1">
   <data>
      <ManagedElement>
         <managedElementId>NodeName</managedElementId>
         <NCTest>
            <nCTestId>1</nCTestId>
            <ActionMoc>
               <actionMocId>1</actionMocId>
               <actionName>
                  <returnValue struct="MyReturnedStruct">
                     <structMember1>42</structMember1>
                     <structMember2>ABC</structMember1>
                     <emptyStructMember/>
                  </returnValue>
               </actionName>
            </ActionMoc>
         </NCTest>
      </ManagedElement>
   </data>
</rpc-reply>
]]>]]>
```

Example 147 Struct Return Value

Error indications are shown in Example 148, Example 149, and Example 150.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>resource-denied</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Action booleanAdder
      MOC ActionMoc failed, missing param param2</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 148 Error Indication on Missing Action Parameter

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>unknown-attribute</error-tag>
      <error-severity>error</error-severity>
      <error-info>
         <bad-attribute>invalidParam</bad-attribute>
         <bad-element>booleanAdder</bad-element>
      </error-info>
      <error-message xml:lang="en">ManagedElement=NodeName,
      NCTest=1,ActionMoc=1</error-message>
   </rpc-error>
</rpc-reply>
```

Example 149 Error Indication on Invalid Action Parameter

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <rpc-error>
      <error-type>application</error-type>
      <error-tag>data-missing</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">MO: ManagedElement=
      NodeName,SystemFunctions=1,File=1,Logical=1,File=MyFile
      is not available (no instance).</error-message>
   </rpc-error>
</rpc-reply>
]]>]]>
```

Example 150 Error Indication on Non-Existing MO

12.5 Modifications to Existing Operations

None.

13 <notification> Capability

This section describes the COM NETCONF-specific `:notification` capability. The description is based on the standard capability template defined in RFC 6241 – NETCONF Configuration Protocol.

13.1 Overview

This capability introduces new notification types and filter elements for `<create-subscription>` operation.

13.2 Dependencies

None.

13.3 Capability Identifier

The notification capability is identified by the following capability string:

`urn:ericsson:com:netconf:notification:1.0`

13.4 New Operations

None.

13.5 Modifications to Existing Operations

This section describes the modifications to the existing operations.

13.5.1 <create-subscription>

As an extension, COM NETCONF-specific notification names can be present in the filter parameter of the `<create-subscription>` operation, as follows:

- The standard defined `<filter>` parameter of `<create-subscription>` operation contains zero or more COM-specific `<event>` XML elements.
- An `<event>` contains exactly one `<filterType>` element. Valid values of `objectCreated`, `objectDeleted`, and `attributeChanged` means that subscription is requested to `<objectCreated>`, `<objectDeleted>`, or `<AVC>` notifications, respectively.

- An <event> contains exactly one <filterValue> element that contains the DN of interest. The name can hold regular expressions in accordance to the POSIX® Extended Regular Expressions syntax.

- The namespace of the COM-specific <event>, <filterType>, and <filterValue> elements is "urn:ericsson:com:netconf:notification:1.0". However, the presence of this namespace is optional and its value is ignored by the COM NETCONF server.

13.6 New Notifications

The following COM NETCONF-specific notifications are supported:

- <replayTimeUnsupported> and <CMSynchronizationRecommended> notifications indicating subscription-related events. These notifications cannot be filtered out and are always sent to the client that has a subscription.

- <objectCreated>, <objectDeleted>, and <AVC> notifications indicating MO changes.

- <heartbeat> notifications are periodic notifications sent to the subscriber, if it has enabled heartbeat capability in the session.

13.6.1 <CMSynchronizationRecommended> Notification

Notification <CMSynchronizationRecommended> (see Example 151) indicates that one or more MO change-related notifications could not be sent to the subscribed client and for this reason, synchronization of the configuration and state data is recommended by either of the following ways:

- Resending of the lost notifications can be requested by the notification replay request, see Section 7.5.1 Create Subscription for Notification Replay on page 71.

As an example, the notification is triggered by the following cases:

- System outage

- Notification buffer overflow

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Add further product-specific event that results in event loss.

The client that receives this notification must retrieve information on the actual system state and configuration by operations <get> and <get-config>.

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2011-09-29T00:02:00+02:00</eventTime>
   <CMSynchronizationRecommended
   xmlns="urn:ericsson:com:netconf:notification:1.0"/>
</notification>
```

Example 151 <CMSynchronizationRecommended>

13.6.2 <replayTimeUnsupported> Notification

The COM-specific notification `<replayTimeUnsupported>`, see Example 152, is sent if `<startTime>` of the subscription is earlier than the earliest available notification in the replay buffer. After this notification is sent, the replay begins with the earliest available notification.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2012-11-14T21:54:59+01:00</eventTime>
   <replayTimeUnsupported
   xmlns="urn:ericsson:com:netconf:notification:1.0"/>
</notification>
]]>]]>
```

Example 152 <replayTimeUnsupported>

13.6.3 MO Change Notifications

To keep the NETCONF server and the client aligned, the following MO change notifications are provided:

- `<objectCreated>`, see Section 13.6.3.2 <objectCreated> Notification on page 116
- `<objectDeleted>`, see Section 13.6.3.1 <objectDeleted> Notification on page 116
- `<AVC>` (Attribute Value Change), see Section 13.6.3.3 <AVC> Notification on page 117

These notifications are sent in a session if the following conditions are met:

- The user of the session has read privileges for the changed MO or attribute.
- The session has subscription with matching filter type and value.
- The change is applied, that is, the transaction that contains the change request has been committed in case of configuration data.
- The change has occurred between the subscription start and end time.

The notification is sent to the client if these conditions are met independently from the fact that the client initiated the change or not.

The MO change notifications contain the COM NETCONF-specific <event> element that encloses the <objectCreated>, <objectDeleted>, and <AVC> elements that indicate the type of change.

Limit on the size of the event notification and length of the event fields are not enforced.

13.6.3.1 <objectDeleted> Notification

Notification <objectDeleted> is sent if an MO is deleted.

The DN of the deleted MO is present in XML attribute dn, as shown in Example 153.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2011-11-24T15:52:46+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <objectDeleted dn="ManagedElement=1,TestRootMoc=1,BasicThing=2"/>
   </events>
</notification>
]]>]]>
```

Example 153 <objectDeleted>

13.6.3.2 <objectCreated> Notification

Notification <objectCreated> is sent if an MO is created.

Note: **INSTRUCTION FOR DOCUMENT REUSE:** If the implementation does not support MafOamSpiCmEvent SPI, then the limitation is: ‘‘As a limitation, MO creations performed in COM NBI (NETCONF or CLI) are notified, and not the ones that are created by the system or created in other interfaces.’’

The DN of the created MO is present in XML attribute dn, as shown in Example 154.

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2011-11-24T15:48:14+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <objectCreated dn="ManagedElement=NodeName,TestRootMoc=1,
      BasicThing=2"/>
   </events>
</notification>
```

Example 154 <objectCreated>

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Depending on SA implementation, either of the alternative is supported. Remove the alternative that is not supported.

Alternative 1: Notification <objectCreated> is followed by <AVC> notifications on attribute values of the created MO.

Alternative 2: Notification <objectCreated> contains the attribute values of the created MO.

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2011-11-24T15:48:14+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <objectCreated dn="ManagedElement=NodeName,TestRootMoc=1,
      BasicThing=2"/>
         <attr name="aStringAttr">
            <v>ABC</v>
         </attr>
         <attr name="anInt8Attr">
            <v>42</v>
         </attr>
   </events>
</notification>
```

Example 155 <objectCreated>

13.6.3.3 <AVC> Notification

Notification `<AVC>` is sent if the value of an attribute is changed. No `<AVC>` notification is sent if the MO attribute is defined with property `noNotification`.

Element `<events>` contains one `<AVC>` element with the following content:

- Element `<AVC>` has one `dn` attribute that contains the DN of the changed MO.
- Element `<AVC>` contains one `<attr>` element.
- Element `<attr>` has one `name` attribute that contains the name of the changed attribute.
- The attribute value after the change is contained in element `<v>`. Element `<attr>` contains one `<v>` element for one attribute value. That is, if the attribute is defined as `sequence` (multivalue) and has multiple values set, then multiple `<v>` elements are present. All the values of a `sequence` attribute are sent in the notification, not only the changed ones.
- If the value of the MO attribute is a `struct`, the `v` tag contains a sequence of `elem` elements, each one with the name of the struct member. Each `elem` tag contains at least one `v` tag that contains the value of the struct member, the same way as for attribute members. When `struct` is created or modified through NBI (CLI or NETCONF), all the `struct` members are sent in the notification.

Examples of `<AVC>` notifications are shown in Example 156 through Example 162.

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2011-11-24T15:48:14+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <AVC dn="ManagedElement=NodeName,TestRootMoc=1,BasicThing=2">
         <attr name="anInt8Attr">
            <v>0</v>
         </attr>
      </AVC>
   </events>
</notification>
```

Example 156 AVC of Single-Valued Integer

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2011-11-24T15:48:14+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <AVC dn="ManagedElement=NodeName,TestRootMoc=1">
         <attr name="aBoolAttr">
            <v>false</v>
         </attr>
      </AVC>
   </events>
</notification>
```

Example 157 AVC of Single-Valued Boolean

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2011-11-24T15:48:14+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <AVC dn="ManagedElement=NodeName,TestRootMoc=1">
         <attr name="aStringMultivalueAttr">
            <v>str1</v>
            <v>str2</v>
            <v>str3</v>
         </attr>
      </AVC>
   </events>
</notification>
```

Example 158 AVC of Sequence (Multivalue) Attribute

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2011-11-24T15:48:14+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <AVC dn="ManagedElement=NodeName,TestRootMoc=1">
         <attr name="aSimpleStruct">
            <v>
               <elem name="int1">
                  <v>10</v>
               </elem>
               <elem name="str1">
                  <v>str1</v>
               </elem>
            </v>
         </attr>
      </AVC>
   </events>
</notification>
```

Example 159 AVC of Struct Attribute

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2011-11-24T15:48:14+01:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <AVC dn="ManagedElement=NodeName,TestRootMoc=1">
         <attr name="aSimpleStruct1">
            <v>
               <elem name="e1">
                  <v>1</v>
               </elem>
               <elem name="e2">
                  <v>str1</v>
               </elem>
            </v>
            <v>
               <elem name="e1">
                  <v>2</v>
               </elem>
               <elem name="e2">
                  <v>str2</v>
               </elem>
            </v>
         </attr>
      </AVC>
   </events>
</notification>
```

Example 160 AVC of Sequence of Structs

If an attribute is defined as `nillable` and all its attribute values are deleted, then an `AVC` notification without value element is sent, as shown in Example 161.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2013-04-15T09:53:46+02:00</eventTime>
   <events xmlns="urn:ericsson:com:netconf:notification:1.0">
      <AVC dn="ManagedElement=NodeName">
         <attr name="userLabel"/>
      </AVC>
   </events>
</notification>
]]>]]>
```

Example 161 AVC of Unset Attribute

If an attribute is defined as `isExclusive`, the `struct` member with value displays the value set and the other member is not display any value since it is empty. The AVC notification is shown in Example 162:

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2014-09-18T15:48:14+02:00</eventTime>
  <events xmlns="urn:ericsson:com:netconf:notification:1.0">
    <AVC dn="ManagedElement=NodeName,TestRootMoc=1,CrazyThing=1">
      <attr name="anExclusiveStruct">
        <v>
          <elem name="hostId">
            <v>testId</v>
          </elem>
          <elem name="ipAddress"/>
        </v>
      </attr>
    </AVC>
  </events>
</notification>
```

Example 162 AVC of isExclusive struct Attribute

13.6.4 <heartbeat> Notification

The client receives the heartbeat notification periodically if it has enabled heartbeat capability (through the `hello` message).

Example of `<heartbeat>` notifications is shown in Example 163.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2015-03-09T13:12:05Z</eventTime>
   <heartbeat xmlns="urn:ericsson:com:ericsson:heartbeat:1.0"/>
</notification>
]]>]]>
```

Example 163 <heartbeat>

13.7 XML Schema

The NETCONF message content is described by XML schema allowing the NETCONF client to recognize generic syntax constraints.

For complete lists of message validity constrain, see Section 13.5.1 <create-subscription> on page 113 and Section 13.6 New Notifications on page 114.

13.7.1 Notification XML Schema

For XML schema for COM-specific notifications, refer to *COMNotification.xsd*.

Note: Because of limitations in the XSD language, validation against `COMNotification.xsd` is successful if a `v` tag contains a sequence of `elem` tags and text content between them. However, this situation is not valid. Post-validation processing must be made to resolve this situation.

`COMNotification.xsd` imports schema definitions from RFC 5277 (schema location `notification.xsd` and `notificationmanagement.xsd`) and from RFC 4741 (schema location `netconf-RFC4741.xsd`).

Note: Do not import `netconf.xsd` defined in RFC 6241, as RFC 6241 provides the needed filter element definitions in YANG format, instead of XSD.

13.7.2 Filter Extension XML Schema

For XML schema for COM-specific `<create-subscription>` filter extension, refer to *netconf.xsd*.

`netconf.xsd` imports schema definitions from `COMNotification.xsd`, see Section 13.7.1 Notification XML Schema on page 120.

According to the schema definition, the possible values of the `<filterType>` element are `ObjectCreated`, `ObjectDeleted`, `AttributeChanged`, and all are accepted.

According to the schema definition, the `select` XML attribute can be present in the `<filter>` element, however the attribute is not a valid parameter of the `<create-subscription>` operation.

Note: Presence of namespace of the extension (`xmlns="urn:ericsson:com:netconf:notification:1.0"`) is optional in the `<event>` element, however the namespace must be present according to the XSDs to form a valid XML document.

14 RFC Compliance Latest

Behaviors of some operations, whose missed compliance to RFC is compliant to RFC if this mode is enabled, are as follows:

- `get` operation does not return any error if instance of requested MO does not exist, see Section 7.1.7 <get> Operation with RFC Compliance Latest Mode on page 47.
- Edit-config `create` operation given at parent MO is inherited to the child MO even if default-operation is set to `none`, see Section 7.3.1.2 Create Operation with RFC Compliance Latest Mode on page 54.
- Edit-config `replace` operation given at parent MO is inherited to the child MO, see Section 7.3.5.3 Replace Operation with RFC Compliance Latest Mode on page 65.
- Edit-config `merge` operation is allowed even with missing members in the struct, see Section 9.6.1 Omitting Members in Structs on page 87.

15 NETCONF Statement of Compliancies

This section provides a summary of the compliance to NETCONF Statement of Compliancies (SoCs).

15.1 Compliance to RFC 4741 and RFC 6241

Table 12 provides a summary of the compliance to the following:

- RFC 4741 – NETCONF Configuration Protocol

  RFC 4741 is obsoleted by RFC 6241.

- RFC 6241 – Network Configuration Protocol (NETCONF)

Table 12 RFC 4741 and RFC 6241 SoC

RFC Section	Compliance
1. Introduction	Non-normative
1.1. Terminology(1)	Non-normative
1.2. Protocol Overview	Compliant Optional requirement to support multiple parallel sessions is supported: ‘‘A device MUST support at least one NETCONF session and SHOULD support multiple sessions.’’
1.3. Capabilities	Partially compliant, as models and the action operation are not represented as capabilities
1.4. Separation of Configuration and State Data	Compliant
2. Transport Protocol Requirements	Compliant
2.1. Connection-Oriented Operation	Compliant
2.2. Authentication, Integrity, and Confidentiality	Compliant
2.3. Mandatory Transport Protocol	Compliant
3. XML Considerations	Compliant Optional requirement on UTF-8 enforcement is not supported: ‘‘If a peer receives an `<rpc>` message that is not well-formed XML or not encoded in UTF-8, it SHOULD reply with a "malformed-message" error.’’

Table 12 RFC 4741 and RFC 6241 SoC

RFC Section	Compliance
3.1. Namespace	Compliant
3.2. Document Type Declarations	Compliant
4. RPC Model	Compliant
4.1. <rpc> Element	Compliant
4.2. <rpc-reply> Element	Compliant
4.3. <rpc-error> Element	Compliant The optional function to send an error message in warning conditions is not supported, however this function is marked by the RFC as for future use. The optional requirement to include language attribute in `<error-message>` is supported: ‘‘This element SHOULD include an "xml:lang" attribute as defined in [W3C.REC-xml-20001006] and discussed in [RFC3470].’’
4.4. <ok> Element	Compliant
4.5. Pipelining	Compliant
5. Configuration Model	Compliant
5.1. Configuration Datastores	Compliant
5.1.3. Filtering	Compliant
5.1.3.1. XPath	The optional XPath capability is not supported
5.1.3.2. Subtree filtering	Partially compliant, as indicated for Section 6.
5.2. Data Modeling	Partially compliant As a limitation, COM NETCONF does not use capabilities to announce the set of data models that the Managed Element implements
6. Subtree Filtering	Partially compliant, see details in subsections
6.1. Overview	Compliant
6.2. Subtree Filter Components	Partially compliant, see details in subsections
6.2.1. Namespace Selection	Not compliant, as namespaces are ignored

Table 12 RFC 4741 and RFC 6241 SoC

RFC Section	Compliance
6.2.2. Attribute Match Expressions	Not compliant, as filtering on XML attributes is not supported. The following properties are represented in the data model as XML attributes: namespace, structure name, and unset MO attribute (`unset="true"`), see Section 6 on page 21. MO attributes are modeled as child XML elements.
6.2.3. Containment Nodes	Compliant
6.2.4. Selection Nodes	Compliant
6.2.5. Content Match Nodes	Compliant
6.3. Subtree Filter Processing	Compliant
6.4. Subtree Filtering Examples	Non-normative
6.4.1. No Filter	Non-normative
6.4.2. Empty Filter	Non-normative
6.4.3. Select the Entire <users> Subtree	Non-normative
6.4.4. Select All <name> Elements within the <users> Subtree	Non-normative
6.4.5. One Specific <user> Entry	Non-normative
6.4.6. Specific Elements from a Specific <user> Entry	Non-normative
6.4.7. Multiple Subtrees	Non-normative
6.4.8. Elements with Attribute Naming	Non-normative
7. Protocol Operations	Partially compliant, as indicated in the subsection
7.1. <get–config>	Compliant

Table 12 RFC 4741 and RFC 6241 SoC

RFC Section	Compliance
7.2. <edit–config>	Partially Compliant Compliant to the operation type requirements of RFC 4741. It includes operations `<create>`, `<delete>`, `<replace>`, and `<merge>` Partially compliant to the operation type requirements of RFC 6241, as option `<remove>`, introduced in RFC 6241, is not supported Partially compliant to the namespace requirements of RFC 4741 and RFC 6241, as namespaces in request messages are ignored. (“The contents MUST be placed in an appropriate namespace, to allow the device to detect the appropriate data model, and the contents MUST follow the constraints of that data model, as defined by its capability definition.”) As a limitation, session is closed if `<edit-config>` contains unsupported standard defined XML elements or any invalid XML content. For the changes done by `<edit-config>` the following two commit behaviors are supported: • RFC compliant – commits the changes after each `<edit-config>`. • COM NETCONF legacy – commits the changes only when the `<close-session>` message is issued.
7.3. <copy-config>	Partially compliant Only supports the operation with source as `<running>` and target as `<startup>`, and with source as `<startup>` and target as `<running>`. As a limitation, the session is closed after the error reply on non-supported operation that is sent as response.
7.4. <delete-config>	Compliant
7.5. <lock>	Not compliant

Table 12 RFC 4741 and RFC 6241 SoC

RFC Section	Compliance
7.6. <unlock>	Not compliant
7.7. <get>	Compliant
7.8. <close-session>	Compliant
7.9. <kill-session>	Compliant
8. Capabilities	Partially compliant, as models and the action operation are not represented as capabilities
8.1. Capabilities Exchange	Compliant Capability `:base:1.1` is not present in capability exchange
8.2. Writable-Running Capability	Compliant
8.2.1. Description	Compliant
8.2.2. Dependencies	Compliant
8.2.3. Capability Identifier	Compliant
8.2.4. New Operations	Compliant
8.2.5. Modifications to Existing Operations	Compliant
8.3. Candidate Configuration Capability	Not compliant, optional feature
8.3.1. Description	Not compliant
8.3.2. Dependencies	Not compliant
8.3.3. Capability Identifier	Not compliant
8.3.4. New Operations	Not compliant
8.3.5. Modifications to Existing Operations	Not compliant
8.4. Confirmed Commit Capability	Not compliant, optional feature
8.4.1. Description	Not compliant
8.4.2. Dependencies	Not compliant
8.4.3. Capability Identifier	Not compliant
8.4.4. New Operations	Not compliant
8.4.5. Modifications to Existing Operations	Not compliant
8.5. Rollback–on–Error Capability	Partially compliant, only compliant using “commit after `<edit-config>`” commit behavior.
8.5.1. Description	Partially compliant
8.5.2. Dependencies	Partially compliant
8.5.3. Capability Identifier	Partially compliant

Table 12 RFC 4741 and RFC 6241 SoC

RFC Section	Compliance
8.5.4. New Operations	Partially compliant
8.5.5. Modifications to Existing Operations	Partially compliant
8.6. Validate Capability	Not compliant
8.6.1. Description	Not compliant
8.6.2. Dependencies	Not compliant
8.6.3. Capability Identifier	Not compliant
8.6.4. New Operations	Not compliant
8.6.5. Modifications to Existing Operations(1)	Not compliant
8.7. Distinct Startup Capability	Partially compliant
8.7.1. Description	Compliant
8.7.2. Dependencies	Compliant
8.7.3. Capability Identifier	Compliant
8.7.4. New Operations	Compliant
8.7.5. Modifications to Existing Operations	Partially compliant Only supports to add the `<startup>` configuration datastore to arguments of the `<copy-config>` and `<delete-config>` operations.
8.8. URL Capability	Not compliant
8.8.1. Description	Not compliant
8.8.2. Dependencies	Not compliant
8.8.3. Capability Identifier	Not compliant
8.8.4. New Operations	Not compliant
8.8.5. Modifications to Existing Operations	Not compliant
8.9. XPath Capability	Not compliant
8.9.1. Description	Not compliant
8.9.2. Dependencies	Not compliant
8.9.3. Capability Identifier	Not compliant
8.9.4. New Operations	Not compliant
8.9.5. Modifications to Existing Operations	Not compliant

Table 12 RFC 4741 and RFC 6241 SoC

RFC Section	Compliance
9. Security Considerations	Compliant Optional requirement on authorization is supported: "‘‘mplementors SHOULD provide a comprehensive authorization scheme with NETCONF.’’ Optional requirement on security is supported: ‘‘this protocol SHOULD be implemented carefully with adequate attention to all manner of attack one expect to experience with other management interfaces.’’
10. IANA Considerations	Compliant
10.1. NETCONF XML Namespace	Compliant
10.2. NETCONF XML Schema	Compliant
10.3. NETCONF YANG Module	Partly compliant with limitations indicated for the previous sections
10.4. NETCONF Capability URNs	Compliant
11. Contributors	Non-normative
12. Acknowledgements	Non-normative
13. References	Non-normative
13.1. Normative References	Non-normative
13.2. Informative References	Non-normative
Appendix A. NETCONF Error List	Compliant Optional requirement on ‘‘partial-operation’’ error tag is supported: ‘‘This error-tag is obsolete, and SHOULD NOT be sent by servers conforming to this document.’’
Appendix B. XML Schema for NETCONF Messages Layer	Compliant
Appendix C. YANG Module for NETCONF Protocol Operations(1)	Not compliant
Appendix D. Capability Template(2)	Compliant
D.1. capability-name (template)	Compliant
D.1.1. Overview	Compliant
D.1.2. Dependencies	Compliant
D.1.3. Capability Identifier	Compliant
D.1.4. New Operations	Compliant

Table 12 RFC 4741 and RFC 6241 SoC

RFC Section	Compliance
D.1.5. Modifications to Existing Operations	Compliant
D.1.6. Interactions with Other Capabilities	Compliant
Appendix E. Configuring Multiple Devices with NETCONF(3)	Non-normative
E.1. Operations on Individual Devices	Non-normative
E.1.1. Acquiring the Configuration Lock	Non-normative
E.1.2. Checkpointing the Running Configuration	Non-normative
E.1.3. Loading and Validating the Incoming Configuration(4)	Non-normative
E.1.4. Changing the Running Configuration	Non-normative
E.1.5. Testing the New Configuration	Non-normative
E.1.6. Making the Change Permanent	Non-normative
E.1.7. Releasing the Configuration Lock	Non-normative
E.2. Operations on Multiple Devices	Non-normative
Appendix F. Deferred Features(5)	Non-normative
Appendix G. Changes from RFC 4741(6)	Non-normative

(1) Added in RFC 6241.
(2) Appendix C in RFC 4741.
(3) Appendix D in RFC 4741.
(4) Removed in RFC 6241.
(5) Present in RFC 4741 only.
(6) Present in RFC 6241 only.

15.2 Compliance to RFC 4742 and RFC 6242

Table 13 provides a summary of the compliance to the following:

- RFC 4742 – Using the NETCONF Configuration Protocol over Secure SHell (SSH)

  RFC 4742 is obsoleted by RFC 6242.

- RFC 6242 – Using the NETCONF Protocol over Secure Shell (SSH)

Table 13 RFC 4742 and RFC 6242 SoC

RFC Section	Compliance
1. Introduction	Non-normative

Table 13 RFC 4742 and RFC 6242 SoC

RFC Section	Compliance
2. Requirements Terminology	Non-normative
3. Starting NETCONF over SSH	Partially Compliant Not compliant to the following requirement: "To allow NETCONF traffic to be easily identified and filtered by firewalls and other network devices, NETCONF servers MUST default to providing access to the "NETCONF" SSH subsystem only when the SSH session is established using the IANA-assigned TCP port 830." Optional requirement on configurable port is supported: "Servers SHOULD be configurable to allow access to the NETCONF SSH subsystem over other ports."
3.1. Capabilities Exchange	Compliant
4. Using NETCONF over SSH	Compliant to RFC 4742, as framing `]]>]]>` is supported Partially compliant to RFC 6242, see details in subsections
4.1. Framing Protocol(1)	Compliant to RFC 6242
4.2. Chunked Framing Mechanism(1)	Not compliant to RFC 6242, as the mandatory chunked framing mechanism is not supported
4.3. End-of-Message Framing Mechanism(1)	Compliant to RFC 6242
5. Exiting the NETCONF Subsystem	Compliant
6. Security Considerations	Compliant
7. IANA Considerations	Non-normative
8. Acknowledgements	Non-normative
9. References	Non-normative
9.1. Normative References	Non-normative
9.2. Informative References	Non-normative
Appendix A. Changes from RFC 4742(1)	Non-normative

(1) Added in RFC 6242.

15.3 Compliance to RFC 5277

Table 14 provides a summary of the compliance to RFC 5277 – NETCONF Event Notifications.

Table 14 RFC 5277 SoC

RFC Section	Compliance
1. Introduction	Non-normative
1.1. Definition of Terms	Non-normative
1.2. Motivation	Non-normative
1.3. Event Notifications in NETCONF	Compliant
2. Notification-Related Operations	Compliant
2.1. Subscribing to Receive Event Notifications	Compliant
2.1.1. <create-subscription>	Partially compliant As a limitation, COM NETCONF-specific extensions to the filter elements are supported instead of the standard `<subtree>` filter with event fields
2.2. Sending Event Notifications	Compliant
2.2.1. <notification>	Compliant
2.3. Terminating the Subscription	Compliant
3. Supporting Concepts	Compliant
3.1. Capabilities Exchange	Compliant
3.1.1. Capability Identifier	Compliant
3.1.2. Capability Example	Non-normative
3.2. Event Streams	Compliant The optional notification-logging and replay service is supported
3.2.1. Event Stream Definition	Compliant
3.2.2. Event Stream Content Format	Compliant
3.2.3. Default Event Stream	Compliant Only the default and mandatory `NETCONF` notification event stream is supported
3.2.4. Event Stream Sources	Compliant
3.2.5. Event Stream Discovery	Compliant
3.2.5.1. Name Retrieval Using `<get>` Operation	Partially Compliant As a limitation, COM NETCONF does not support user-specified filters on the `<get>` operation.
3.2.5.2. Event Stream Subscription	Compliant

Table 14 RFC 5277 SoC

RFC Section	Compliance
3.2.5.2.1. Filtering Event Stream Contents	Compliant Only the `<subtree>` filter type is supported
3.3. Notification Replay	Compliant
3.3.1. Overview	Compliant
3.3.2. Creating a Subscription with Replay	Compliant
3.4. Notification Management Schema	Compliant
3.5. Subscriptions Data	Compliant
3.6. Filter Mechanics	Compliant
3.6.1. Filtering	Compliant
3.7. Message Flow	Compliant
4. XML Schema for Event Notifications	Partly compliant As a limitation, COM NETCONF-specific extensions to the filter elements are supported instead of the standard filter elements
5. Filtering Examples	Non-normative
5.1. Subtree Filtering	Non-normative
5.2. XPATH Filters	Non-normative
6. Interleave Capability	Not compliant The interleave capability is optional to support
6.1. Description	Not compliant
6.2. Dependencies	Not compliant
6.3. Capability Identifier	Not compliant
6.4. New Operations	Not compliant
6.5. Modifications to Existing Operations	Not compliant
7. Security Considerations	Compliant
8. IANA Considerations	Compliant
9. Acknowledgements	Non-normative
10. Normative References	Non-normative

15.4 Compliance to RFC 5539

Table 15 provides a summary of the compliance to RFC 5539 – NETCONF over Transport Layer Security (TLS).:

Table 15 RFC 5539 SoC

RFC Section	Compliance
1. Introduction	Compliant
1.1. Conventions Used in This Document	Non-normative
2. NETCONF over TLS	Non-normative
2.1. Connection Initiation	Compliant(1) The mandatory-to-implement cipher suite (`TLS_RSA_WITH_AES_128_CBC_SHA`) is supported (2)
2.2. Connection Closure	Compliant
3. Endpoint Authentication and Identification	Compliant
3.1. Server Identity	Not applicable, as the COM NETCONF server acts as server, thus only client identity is to be verified
3.2. Client Identity	Compliant
4. Security Considerations	Compliant
5. IANA Considerations	Compliant
6. Acknowledgements	Non-normative
7. Contributor's Address	Non-normative
8. References	Non-normative
8.1. Normative References	Non-normative
8.2. Informative References	Non-normative

(1) The compliance is ensured by OpenSSL.
(2) ***INSTRUCTION FOR DOCUMENT REUSE:*** *List the additional product-specific supported cipher suites.*