

# Ericsson NETCONF Interface

---

## INTERWORK DESCRIPTION

**Copyright**

© Ericsson AB 2016, 2017. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

**Disclaimer**

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

**Trademark List**

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Related Information	1
1.2	Key Features of Ericsson NETCONF Interface	1
1.3	Prerequisites	2
<b>2</b>	<b>NETCONF Session</b>	<b>3</b>
2.1	Start a NETCONF Session over SSH	3
2.2	Start a NETCONF Session over TLS	4
2.3	Session Inactivity Timer	5
2.4	Session End	6
<b>3</b>	<b>NETCONF RPC Messages</b>	<b>7</b>
3.1	Message <hello>	7
3.2	Message <ok>	11
3.3	Error Messages	11
<b>4</b>	<b>NETCONF Data Model</b>	<b>15</b>
4.1	Model Definition	15
4.2	Namespaces	15
4.3	Configuration and State Data	16
4.4	Distinguished Names	16
4.5	MO Attributes	17
4.6	MO Instances	22
4.7	MO Actions	23
<b>5</b>	<b>NETCONF Operations</b>	<b>25</b>
5.1	Operation <get>	25
5.2	Operation <get-config>	45
5.3	Operation <edit-config>	46
5.4	Operation <action>	65
5.5	Operation <create-subscription>	65
5.6	Operation <close-session>	73
5.7	Operation <kill-session>	73
5.8	Operation <copy-config>	74
5.9	Operation <delete-config>	76



<b>6</b>	<b>NETCONF Notifications</b>	<b>81</b>
6.1	Notification <replayComplete>	81
6.2	Notification <notificationComplete>	81
6.3	Non-Standard Notifications	82
<b>7</b>	<b>NETCONF Validation</b>	<b>83</b>
7.1	Create MO Instance	83
7.2	Delete MO Instance	84
7.3	Change Single-Valued Attributes	85
7.4	Delete MO Attribute Values	88
7.5	Change Sequence Attributes	89
7.6	Change Struct Attributes	89
7.7	System-Created Property	92
<b>8</b>	<b>NETCONF Visibility Control</b>	<b>105</b>
<b>9</b>	<b>NETCONF Transaction</b>	<b>107</b>
9.1	Transaction Start	107
9.2	Locking	107
9.3	Transaction Commit	108
9.4	Abort	109
9.5	Transaction Time-Out	109
<b>10</b>	<b>NETCONF Capability &lt;action&gt;</b>	<b>111</b>
10.1	Overview	111
10.2	Dependencies	111
10.3	Capability Identifier	111
10.4	New Operations	111
10.5	Modifications to Existing Operations	115
<b>11</b>	<b>NETCONF Capability &lt;notification&gt;</b>	<b>117</b>
11.1	Overview	117
11.2	Dependencies	117
11.3	Capability Identifier	117
11.4	New Operations	117
11.5	Modifications to Existing Operations	117
11.6	New Notifications	118
11.7	XML Schema	127
<b>12</b>	<b>RFC Compliance Latest</b>	<b>129</b>



<b>13</b>	<b>NETCONF Statement of Compliance</b>	<b>131</b>
13.1	NETCONF Capabilities	131
13.2	NETCONF Compliance to RFC 4741 and RFC 6241	133
13.3	NETCONF Compliance to RFC 4742 and RFC 6242	140
13.4	NETCONF Compliance to RFC 5277	142
13.5	NETCONF Compliance to RFC 5539	144





# 1 Introduction

This document describes the functions provided in the Ericsson Network Configuration (NETCONF) interface.

The NETCONF interface is a machine to machine interface for configuration management of the Managed Element (ME) using the NETCONF protocol over the Secure Shell (SSH) or the NETCONF protocol over the Transport Layer Security (TLS).

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

## 1.1 Related Information

For information on the Managed Object Model (MOM), Managed Object Classes (MOCs), Managed Objects (MOs), and related concepts mentioned in this document, refer to *Managed Object Model User Guide*.

## 1.2 Key Features of Ericsson NETCONF Interface

The key features of the Ericsson NETCONF interface are described in Table 1.

*Table 1 Key Features of Ericsson NETCONF Interface*

Feature	Description
Access control	The result of the NETCONF operations is subject of authentication and authorization.
Data model	Configuration and state data is represented as MO instances, attributes, and actions in Ericsson NETCONF-specific XML format in Ericsson NETCONF messages. The Ericsson NETCONF XML is described in Section 4 on page 15.
Datastore	The running and startup datastore are supported based on the operation. For details about the operations, see Section 5 on page 25.
Log	All operations are logged. The log records contain information on username, session ID, operation time, name with parameters, and operation responses.

*Table 1 Key Features of Ericsson NETCONF Interface*

Feature	Description
Model driven	The MO classes, attribute types, and action properties are defined by the information models described in the MOM.  As a limitation, the Ericsson NETCONF interface does not use capabilities to announce the set of data models that the ME implements.
Multiple sessions	Multiple parallel sessions are supported.
Namespaces	The generated NETCONF responses contain an XML namespace, defined by XML attribute <code>xmlns</code> . The namespace is defined in the MOM. However, <code>xmlns</code> is ignored 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 the Secure Socket Layer (SSL) is provided by the SSH daemon or the Transport Layer Security (TLS) proxy component.
Standard compliance	The Ericsson NETCONF interface is compliant to NETCONF standards, as specified in Section 13 on page 131.
Transactions	Configuration changes are applied through atomic transactions. Thus, it is ensured that all or none of the operations are executed. For details, see Section 9 on page 107.
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. For node name, `NODE06ST` is used as an example.

## 1.3 Prerequisites

This section describes the prerequisites, which must be fulfilled before using the Ericsson NETCONF interface.

### 1.3.1 Conditions

The following conditions must apply:

- A standard SSH version 2 client is available.
- TLS 1.2 support for NETCONF over TLS is available.
- The user has IP access to the Operation and Maintenance (O&M) address of the ME.
- The user has a valid O&M user account.
- Suitable firewall settings enable access to the NETCONF interface port.





## 2 NETCONF Session

Both NETCONF over SSH and TLS can be configured to use other ports than the default port used in the examples. If other values are configured, use those values instead.

Multiple sessions can exist at a time. The maximal number of parallel sessions is 255.

### 2.1 Start a NETCONF Session over SSH

To start a NETCONF session over SSH:

1. Start an SSH session:

Example of logon with OpenSSH client:

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

or

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

The options are as follows:

- **<user>** – Name of a valid user account.
- **<target\_host>** – O&M IP address or hostname of the ME.
- **-p (port number)** – TCP port 830 is default.

Root user access is denied.

2. Wait for the session to start.



**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 the TCP keepalive feature, which notifies peers on connection loss. The feature is optional, it is disabled by default, and it is controlled by the following parameters:

- `Keepalive time` – Duration between two keepalive transmissions in idle condition. In most implementations, the default value is 7200 seconds and the recommended value is 300 seconds.
- `Keepalive interval` – Duration between two successive keepalive retransmissions, if acknowledgment to the previous keepalive transmission is not received. The recommended value is 75 seconds.
- `Keepalive retry` – Number of retransmissions to be done before declaring that the remote end is unavailable. The recommended value is 9.

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

### 2.1.1 Successful Session Start

When the session is established, peers must initiate the capabilities exchange, as described in Section 3.1 Message <hello> on page 7.

### 2.1.2 Unsuccessful Session Start

The session start can fail. For examples of error messages, see Table 2.

*Table 2 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.

## 2.2 Start a NETCONF Session over TLS

To start a NETCONF session over TLS:



## 1. Start a TLS connection.

The options are as follows:

- `<host>` – O&M IP address or hostname of the ME.
- `<port>` – Transmission Control Protocol (TCP) port 6513 is default.
- `<nodecert>` – Certificate file to use, PEM format is assumed.
- `<nodekey>` – The private key to use. If not specified, the certificate file is used.
- `<cacert>` – PEM format file of CAS.

Example command to open NETCONF over TLS connection with `s_client`:

```
openssl s_client -connect <host>:<port> -quiet -tls1  
-bugs -cert <nodecert> -key <nodekey> -CAfile <cacert>
```

## 2. Wait for the session to start.

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

### 2.2.1 Successful Session Start

When the session is established, peers must initiate the capabilities exchange, as described in Section 3.1 Message `<hello>` on page 7.

### 2.2.2 Unsuccessful Session Start

For error messages, refer to product-specific documentation.

## 2.3 Session Inactivity Timer

When the predefined time has passed without activity in the NETCONF interface, the system automatically ends the ongoing session without any warning. Activity in a NETCONF session means any operation resulting in data exchange between the client and the server.

The default inactivity timer value is 5 minutes.

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



## 2.4 Session End

A NETCONF session is closed in the following cases:

- As a result of operation `<close-session>` or `<kill-session>`
- 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 are terminated if attribute *administrativeState* in MO *NetconfTls* is being set to `LOCKED` and is committed
- All the ongoing SSH over NETCONF sessions are closed if *administrativeState* in MO *NetconfSsh* is set to `LOCKED` and is committed (*sshdManagement* in *libcom\_sshd\_manager.cfg* is to be `TRUE`).



## 3 NETCONF RPC Messages

NETCONF messages are encoded in XML and exchanged on top of a simple RPC-based communication model with the elements described in Table 3.

*Table 3 NETCONF RPC Messages*

RPC Message	Description
<code>&lt;hello&gt;</code>	The NETCONF peers send <code>&lt;hello&gt;</code> messages to exchange capability information. For details, see Section 3.1 Message <code>&lt;hello&gt;</code> on page 7.
<code>&lt;rpc&gt;</code>	The NETCONF client sends an <code>&lt;rpc&gt;</code> message to the server as operation request containing <code>message-id</code> . The <code>message-id</code> is an integer monotonically increased during the lifetime of the NETCONF server. For details on NETCONF operation request messages, see Section 5 on page 25.
<code>&lt;rpc-reply&gt;</code>	<p>As a response to the operation request, the NETCONF server sends a single <code>&lt;rpc-reply&gt;</code> message. It contains the <code>message-id</code> from the request and the following elements, depending on the message type:</p> <ul style="list-style-type: none"> <li>On operation success, element <code>&lt;rpc-reply&gt;</code> contains either element <code>&lt;ok&gt;</code> (for details, see Section 3.2 Message <code>&lt;ok&gt;</code> on page 11) or operation-specific elements according to the operation return value definition (for details, see Section 5 on page 25).</li> <li>Otherwise, element <code>&lt;rpc-reply&gt;</code> contains element <code>&lt;rpc-error&gt;</code> if the operation is not completed with an error (for details, see Section 3.3 Error Messages on page 11).</li> </ul>
<code>&lt;notification&gt;</code>	<p>The NETCONF server sends <code>&lt;notification&gt;</code> messages to the subscribed clients in the following cases:</p> <ul style="list-style-type: none"> <li>An event occurs that complies to the notification trigger condition.</li> <li>The client requests notification replay.</li> </ul> <p>For details on NETCONF notifications, see Section 6 on page 81.</p>
<code>]]&gt;]]&gt;</code>	Message termination sequence <code>]]&gt;]]&gt;</code> is used to provide message framing.

### 3.1 Message `<hello>`

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

As shown in Example 1, message `<hello>` sent by the server contains the list of supported capabilities and `<session-id>`, which is an integer increased by new session starts.

```
<?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>
    <capability>urn:com:ericsson:ebase:1.1.0</capability>
    <capability>urn:com:ericsson:ebase:1.2.0</capability>
    <capability>urn:com:ericsson:ebase:2.0.0</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:interleave:1.0</capability>
    <capability>urn:com:ericsson:netconf:notifications-aggregation:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:startup:1.0</capability>
  </capabilities>
  <session-id>1</session-id>
</hello>
]]>]]>
```

#### Example 1 Message `<hello>` Sent by Server

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

As shown in Example 2, message `<hello>` sent by the client must not contain `<session-id>` but the capability list with at least capability `:base`.

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

#### Example 2 Message `<hello>` 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 the MO *Schema* by operation `<get>`.

### 3.1.1 ebase Numbering

The ebase version is numbered as `ebase:X.Y.Z`, where:

- `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 is added for NETCONF expressions that was not previously supported. 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.

### 3.1.2 Capability Identifier ebase

In line with the RFC, the highest common urn:com:ericsson:ebase capability version from the client and server message <hello> determine the server behavior for the session, see Example 3. If no ebase capabilities are advertised by the client, the default behavior set in the sshd configuration is applied.

In Example 3, the highest common capability is urn:com:ericsson:ebase:1.1.0, so the behavior is according to 1.1.0 column in the Table 4.

```

Server 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>
    <capability>urn:com:ericsson:ebase:1.1.0</capability>
    <capability>urn:com:ericsson:ebase:1.2.0</capability>
    <capability>urn:com:ericsson:ebase:2.0.0</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:interleave:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:startup: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>
    <capability>urn:com:ericsson:ebase:1.1.0</capability>
    <capability>urn:ericsson:com:netconf:heartbeat:1.0</capability>
  </capabilities>
</hello>]]>]]>

```

#### Example 3 Determining ebase Version

Multiple behaviors are connected to each ebase version, see Table 4 for the exact pairings and the following sections for explanation. The behaviors connected to the higher ebase versions are considered more RFC-compliant than the ones connected to the lower versions.

**Table 4** Interpretation of ebase Versions

		ebase version			
		0.1.0	1.1.0	1.2.0	2.0.0
Commit at	<close-session>	Yes	No	No	No
	<edit-config>	No	Yes	Yes	Yes
	<action>	No	No	No	Yes
	NEM (NETCONF Extension Module) operation	No	No	No	Yes
Inherit operation from parent		No	No	Yes	Yes
Replace operation deletes MO		No	No	Yes	Yes
Singleton struct merging		No	No	Yes	Yes
Empty <get> result produces error		No	No	Yes	Yes

### 3.1.2.1

#### Session Commit Behavior

In NETCONF, there is an underlying database transaction for the changes done by the requests, see Section 9 on page 107. According to the behavior ebase:0.1.0, the changes made in the transaction are only committed when a <close-session> operation arrives. From ebase:1.1.0, the changes are committed after each <edit-config> operation. From ebase:2.0.0, the changes are committed after each operation that can update the database, those are <edit-config>, <action> and NEM operations. For more information, see Section 9.3 Transaction Commit on page 108.

### 3.1.2.2

#### <edit-config> Operation Attribute

The following are multiple unconnected behavior options related to the operation XML attribute in <edit-config>:

- Inherit operation from parent.

According to the behavior ebase:0.1.0, if there is no operation on a certain MO level, the one specified in the <default-operation> is used. From ebase:1.2.0, if there is a parent MO available, the parent operation is used instead. For more information, see Section 5.3.2.2 Operation create with RFC Compliance Latest Mode on page 50.

- Replace operation delete MO.





According to the behavior ebase:0.1.0, the replace operation attribute on an MO is interpreted as a merge extended with the default values of unspecified attributes, thus the children of the replaced MO are not affected. From ebase:1.2.0, replace is interpreted as deleting the marked MO and recreating it with the given contents; the children of the replaced MO are also deleted and only recreated if they are specified in the request. For more information, see Section 5.3.6.3 Operation replace with RFC Compliance Latest Mode on page 60.

- Merge operation on structs.

RFC does not differentiate between structs and MOs. According to the behavior ebase:0.1.0, merging only certain members of the structs and leaving other members unaffected is not possible. From ebase:1.2.0, singleton structs can be merged similarly to MOs. For more information, see Section 7.6.1 Omitting Members in Structs on page 90.

### 3.1.2.3 Empty <get> or <get-config> Result

According to the behavior ebase:0.1.0, if the <get> or <get-config> operation does have a non-empty <filter> but nothing in the database matches the filter, the reply is an <rpc-error>. From ebase:1.2.0, in such cases an empty <data> is returned. For more information, see Section 5.1.6 Operation <get> with RFC Compliance Latest Mode on page 44.

## 3.2 Message <ok>

Operation completion without error is indicated by message <ok> if the operation has no return value, see Example 4.

```
<?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 4 Message <ok>

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

## 3.3 Error Messages

Any incomplete termination of the operation is indicated by a standard error message, see Example 5 through Example 9.

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



```
<?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-failed</error-tag>
    <error-severity>error</error-severity>
    <error-message xml:lang="en">Malformed XML!!!</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

#### **Example 5** 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=NODE06ST,⇒
SystemFunctions=1,SysM=1,Snmp=3 is not available (no instance).</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

#### **Example 6** Error Message for MO Instance Not Available

The error message returned to the MO creation request on a system-created MO is shown in Example 7.

```
<?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⇒
NODE06ST. MO class ManagedElement, is system created</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

#### **Example 7** Error Message for System-Created MO

```
<?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 8** 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 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>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 9 Error Message for Setting Create as Default Operation***





## 4 NETCONF Data Model

This section describes the Ericsson NETCONF XML format in terms of layout, containment, keying, lookup, replacement, management of data, and its relationship to the information models. Any other constraints imposed by the data model are also described.

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.

### 4.1 Model Definition

The Ericsson NETCONF XML elements represent MO instances and, MO attributes, and MO actions as defined by the information models that are described in the MOM.

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

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

### 4.2 Namespaces

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

- `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, the namespace is ignored by the server, as the MO DN by itself uniquely identifies the MO instance and its class in the ME.

## 4.3 Configuration and State Data

According to the NETCONF principles, the datastore content is separated into configuration data and state data.

### 4.3.1 Configuration Data

Configuration data is the 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.

### 4.3.2 State Data

State data is extra 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.

## 4.4 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 containing 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.

The DN `ManagedElement=NODE06ST` is represented in Example 10.

```
<ManagedElement>  
  <managedElementId>NODE06ST</managedElementId>  
</ManagedElement>
```

*Example 10 DN of ManagedElement*



The DN ManagedElement=NODE06ST, SystemFunctions=1, Fm=1 is represented in Example 11.

```
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
  <managedElementId>NODE06ST</managedElementId>
  <SystemFunctions>
    <systemFunctionsId>1</systemFunctionsId>
    <Fm xmlns="urn:com:ericsson:ecim:ComFm">
      <fmId>1</fmId>
    </Fm>
  </SystemFunctions>
</ManagedElement>
```

**Example 11** DN of MO Fm in Request and Response Messages

As namespaces are optional in request messages, the form shown in Example 12 is also valid.

```
<ManagedElement>
  <managedElementId>NODE06ST</managedElementId>
  <SystemFunctions>
    <systemFunctionsId>1</systemFunctionsId>
    <Fm>
      <fmId>1</fmId>
    </Fm>
  </SystemFunctions>
</ManagedElement>
```

**Example 12** DN of MO 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. For a full list of illegal characters, refer to the 3GPP standard for MO Name convention, 3GPP TS 32.300 V9.0.0.

Illegal characters can be used if they are provided in their hexadecimal representation and then 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 is shown in Example 13.

```
<ManagedElement>
  <managedElementId>NODE06ST</managedElementId>
  <SystemFunctions>
    <systemFunctionsId>1</systemFunctionsId>
    <MoX>
      <moXId>keyWithAPlusCharacter\2B</moXId>
    </MoX>
  </SystemFunctions>
</ManagedElement>
```

**Example 13** Illegal + Character

## 4.5 MO Attributes

The 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.

### 4.5.1 MO Key-Attribute

The key-attribute modeling element is a MOC attribute. Key-attributes are represented as XML elements. The first letter is in lower-case, followed by the string `Id`. The key-attribute value is provided between the opening and closing elements, for example, `<managedElementId>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 Relative Distinguished Name (RDN) value of the MO instance it is contained in. The RDN value space is defined in 3GPP TS 32.300 V9.0.0.

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

The DN `ManagedElement=NODE06ST,NCTest=1,XThing=LOCKED` is represented in Example 14.

```
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
  <managedElementId>NODE06ST</managedElementId>
  <NCTest>
    <nCTestId>1</nCTestId>
    <XThing xc:operation="create">
      <xThingId>LOCKED</xThingId>
    </XThing>
  </NCTest>
</ManagedElement>
```

#### *Example 14 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.

The DN `ManagedElement=NODE06ST,NCTest=1,KThing=1` is represented in Example 15.

```
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
  <managedElementId>NODE06ST</managedElementId>
  <NCTest>
    <nCTestId>1</nCTestId>
    <KThing xc:operation="create">
      <kThingId>1</kThingId>
    </KThing>
  </NCTest>
</ManagedElement>
```

#### *Example 15 DN of KThing in Request Messages*

The `DerivedInteger` type input and the NETCONF interpretation for some special cases are shown in Table 5.





**Table 5** *DerivedInteger Type Input and NETCONF Interpretation*

Input (DerivedInteger)	Interpretation
+1	1
+000000001	1
-1	-1
-0000000000000000000000001	-1
+0000000000000000000000001	1
+000000000000000000000000	0
-000000000000000000000000	0

The negative responses for the key-attributes of the enumeration and integer data types are shown in Example 16 through Example 18.

```
<?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 16** *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 hiii for an integer type attribute =>
kThingId</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

**Example 17** *Error Message for Invalid KeyAttribute Value of EcimDerivedInteger*

An error message for an integer value outside the allowed range, specified in the MOM, is shown in Example 18.

```
<?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 18** Error Message for Integer Value Not Allowed Range

## 4.5.2 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>value</userLabel>`, see Example 19 and Example 20.

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 (`&lt;` and `&#x41;`) is supported.
- Enumeration 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=NODE06ST, SystemFunctions=1, FileM=1, LogicalFs=1, FileGroup=1</fileGroup>`.

The referred MO can be internal or external. .

- Double value is represented as shown in attribute `distanceFactor`. The double data type can be presented both in canonical form and scientific notation representation.

A single-valued attribute (`administrativeState`) represented in the context of the DN (MOC and key attributes) is shown in Example 19.



```
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
  <managedElementId>NODE06ST</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 19 Single-Valued Attribute

```
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
  <managedElementId>NODE06ST</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 20 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 MOs.
- A message with unset="true" is returned if the <get> target is the attribute itself, see Example 21.

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>NODE06ST</managedElementId>
      <userLabel unset="true"></userLabel>
    </ManagedElement>
  </data>
</rpc-reply>
]]>]]>
```

### Example 21 Attribute without Value in Response Message

### 4.5.3 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.

### 4.5.4 Structures

Attributes defined as structure are represented as XML elements enclosing member 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 22. However, 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 member attributes are represented in the same way as attributes of the same data type. A structure member cannot be of struct type.

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

*Example 22 Structure Attribute*

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

*Example 23 Structure Attribute without Structure Name*

## 4.6 MO Instances

MO instances are represented as combined elements of the MO DNs and their attributes, see Example 24.



```
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
  <managedElementId>NODE06ST</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 24** *Snmp and Contained SnmpTargetV2C MO Instance*

## 4.7 MO Actions

For information on MO actions, see Section 10 on page 111.





## 5 NETCONF Operations

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

- `<get>`, see Section 5.1 Operation `<get>` on page 25
- `<get-config>`, see Section 5.2 Operation `<get-config>` on page 45
- `<edit-config>`, see Section 5.3 Operation `<edit-config>` on page 46
- `<action>`, see Section 5.4 Operation `<action>` on page 65
- `<create-subscription>`, see Section 5.5 Operation `<create-subscription>` on page 65
- `<close-session>`, see Section 5.6 Operation `<close-session>` on page 73
- `<kill-session>`, see Section 5.7 Operation `<kill-session>` on page 73
- `<copy-config>`, see Section 5.8 Operation `<copy-config>` on page 74
- `<delete-config>`, see Section 5.9 Operation `<delete-config>` on page 76

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

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

These two operation requests are always replied by the error message shown in Example 25.

```
<?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 25 Error Message for Unsupported Operations*

### 5.1 Operation `<get>`

Operation `<get>` retrieves the configuration and state data from the running datastore.

## Parameter

Operation `<get>` has a single parameter, `<filter>`, with the following properties:

- 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.
- Can contain element `<type>` . If this element is present, it must contain option `subtree`, which is the only supported filter type.
- Can contain filter expressions, as described in Section 5.1.2 Filter for Operation `<get>` on page 27.
- Notification stream discovery can be requested by a special filter construct, as described in Section 5.1.3 Request Event Stream Discovery on page 37.

Option “xpath” in element `<type>` is invalid, as the optional XPath capability is not supported.

## Positive Response

If the operation is completed without error, `<rpc-reply>` is sent. The `<data>` section contains the appropriate subset of configuration and state data in the Ericsson NETCONF XML format, described in Section 4 on page 15.

## Negative Response

If the operation cannot be completed, element `<rpc-error>` is included in `<rpc-reply>`.

As a special case, both `<data>` and `<rpc-error>` elements are present in `<rpc-reply>` if the response sending is started but cannot be completed, see Section 5.1.1 Error during Operation `<get>` Response Processing on page 26.

### 5.1.1 Error during Operation `<get>` Response Processing

If the NETCONF server detects an error during operation `<get>` response processing, the following apply, as shown in Example 26:

- A partial response is provided.
- Element `<data>` in `<rpc-reply>` is closed.
- Message `<rpc-error>` is returned.





```
<?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>NODE06ST</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=NODE06ST, attribute name: productIdentity</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

*Example 26 Error Message for Operation <get>*

#### 5.1.1.1 Error with Source Set to Startup Datastore

Operation <get> retrieves the configuration and runstate data from the running datastore. If the startup datastore is set as the source datastore, the RPC error message in Example 27 is returned.

```
<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 27 Error Message When Startup Datastore is Used*

#### 5.1.2 Filter for Operation <get>

Optionally, both the <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, as described in Section 4 on page 15.

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 the Ericsson NETCONF XML in XML attributes but in XML elements.
- Containment Nodes, Selection Nodes, and Content Match Nodes are supported as described in Section 5.1.4 Support for Nodes on page 38.

### 5.1.2.1 No Filter

The complete content of the datastore is returned if element `<filter>` is not present, see Example 28.

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

*Example 28 Operation `<get>` Request without Filter*

### 5.1.2.2 Empty Filter

A filter with an empty value matches no element as required by the standard, see Example 29 and Example 30.

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

*Example 29 Operation `<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 30 Reply Message*

### 5.1.2.3 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 31.

```
<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>NODE06ST</managedElementId>
        <SystemFunctions>
          <systemFunctionsId>1</systemFunctionsId>
          <Fm/>
        </SystemFunctions>
      </ManagedElement>
    </filter>
  </get>
</rpc>]]>]]>
```

*Example 31 Filtered Operation `<get>` on MO `Fm`*



```
<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>
      <SystemFunctions>
        <systemFunctionsId>1</systemFunctionsId>
        <Fm xmlns="urn:com:ericsson:ecim:ComFm">
          <fmId>1</fmId>
          <lastSequenceNo>40</lastSequenceNo>
          <sumCritical>0</sumCritical>
          <sumMajor>1</sumMajor>
          <sumMinor>0</sumMinor>
          <sumWarning>0</sumWarning>
          <totalActive>1</totalActive>
          <lastChanged>2017-04-20T11:16:36.550+02:00</lastChanged>
          <heartbeatInterval>60</heartbeatInterval>
          <FmAlarm>
            <fmAlarmId>1</fmAlarmId>
            <source>ManagedElement=1, SystemFunctions=1, Fm=1, FmAlarmModel=FileManagement, =>
              FmAlarmType=FileMNoOfFilesExceeded</source>
            <lastEventTime>2017-04-20T10:59:11.137+02:00</lastEventTime>
            <sequenceNumber>36</sequenceNumber>
            <activeSeverity>CRITICAL</activeSeverity>
            <additionalText>The number of files exceeded threshold level <thresholdHigh>=>
              in ManagedElement=1, SystemFunctions=1, FileM=1, LogicalFs=1, FileGroup=1 =>
            </additionalText>
            <majorType>193</majorType>
            <minorType>131073</minorType>
            <specificProblem>File Management, Number of Files in FileGroup Exceeded =>
            </specificProblem>
            <eventType>Other</eventType>
            <probableCause>343</probableCause>
            <originalEventTime>2017-04-20T10:59:11.137+02:00</originalEventTime>
            <originalSeverity>CRITICAL</originalSeverity>
            <originalAdditionalText>The number of files exceeded threshold level <thresholdHigh>=>
              in ManagedElement=1, SystemFunctions=1, FileM=1, LogicalFs=1, FileGroup=1
            </originalAdditionalText>
          </FmAlarm>
          <FmAlarmModel>
            <fmAlarmModelId>FileManagement</fmAlarmModelId>
            <FmAlarmType>
              <fmAlarmTypeId>FileMMaxSizeExceeded</fmAlarmTypeId>
              <majorType>193</majorType>
              <minorType>131074</minorType>
              <moClasses>FileGroup</moClasses>
              <specificProblem>File Management, Max Size in FileGroup Exceeded =>
              </specificProblem>
              <eventType>OTHER</eventType>
              <probableCause>151</probableCause>
              <isStateful>true</isStateful>
              <additionalText></additionalText>
            </FmAlarmType>
            <FmAlarmType>
              <fmAlarmTypeId>FileMNoOfFilesExceeded</fmAlarmTypeId>
              <majorType>193</majorType>
              <minorType>131073</minorType>
              <moClasses>FileGroup</moClasses>
              <specificProblem>File Management, Number of Files in FileGroup =>
                Exceeded</specificProblem>
              <eventType>OTHER</eventType>
              <probableCause>343</probableCause>
              <isStateful>true</isStateful>
              <additionalText></additionalText>
            </FmAlarmType>
          </FmAlarmModel>
        </Fm>
      </SystemFunctions>
    </ManagedElement>
  </data>
</rpc-reply>
```



```
<FmAlarmModel>
  <fmAlarmModelId>CW</fmAlarmModelId>
  <FmAlarmType>
    <fmAlarmTypeId>ComSaCLMClusterNodeUnavailable</fmAlarmTypeId>
    <majorType>193</majorType>
    <minorType>849346561</minorType>
    <moClasses>Amf</moClasses>
    <specificProblem>COM SA, CLM Cluster Node Unavailable =>
    </specificProblem>
    <eventType>PROCESSINGERRORALARM</eventType>
    <probableCause>418</probableCause>
    <isStateful>true</isStateful>
    <additionalText></additionalText>
  </FmAlarmType>
</FmAlarmModel>
<FmAlarmModel>
  <fmAlarmModelId>MDF</fmAlarmModelId>
  <FmAlarmType>
    <fmAlarmTypeId>ComSaMDFDetectedModelError</fmAlarmTypeId>
    <majorType>193</majorType>
    <minorType>849346562</minorType>
    <moClasses>Amf</moClasses>
    <specificProblem>COM SA, MDF Detected Model Error</specificProblem>
    <eventType>PROCESSINGERRORALARM</eventType>
    <probableCause>418</probableCause>
    <isStateful>true</isStateful>
    <additionalText></additionalText>
  </FmAlarmType>
</FmAlarmModel>
<FmAlarmModel>
  <fmAlarmModelId>SwM</fmAlarmModelId>
  <FmAlarmType>
    <fmAlarmTypeId>FallbackOperationStartingSoon</fmAlarmTypeId>
    <majorType>193</majorType>
    <minorType>65537</minorType>
    <moClasses>SwM</moClasses>
    <specificProblem>A Fallback Operation will soon be started =>
    </specificProblem>
    <eventType>OTHER</eventType>
    <probableCause>164</probableCause>
    <isStateful>true</isStateful>
    <additionalText></additionalText>
  </FmAlarmType>
</FmAlarmModel>
<FmAlarmModel>
  <fmAlarmModelId>coreMw</fmAlarmModelId>
  <FmAlarmType>
    <fmAlarmTypeId>ComSaAmfSiUnassigned</fmAlarmTypeId>
    <majorType>18568</majorType>
    <minorType>131077</minorType>
    <moClasses>Amf</moClasses>
    <specificProblem>COM SA, AMF SI Unassigned</specificProblem>
    <eventType>PROCESSINGERRORALARM</eventType>
    <probableCause>418</probableCause>
    <isStateful>true</isStateful>
    <additionalText></additionalText>
  </FmAlarmType>
  <FmAlarmType>
    <fmAlarmTypeId>ComSaAmfComponentCleanupFailed</fmAlarmTypeId>
    <majorType>18568</majorType>
    <minorType>131075</minorType>
    <moClasses>Amf</moClasses>
    <specificProblem>COM SA, AMF Component Cleanup Failed =>
    </specificProblem>
    <eventType>PROCESSINGERRORALARM</eventType>
    <probableCause>418</probableCause>
    <isStateful>true</isStateful>
    <additionalText></additionalText>
  </FmAlarmType>
</FmAlarmModel>
```



```

    <FmAlarmType>
      <fmAlarmTypeId>ComSaAmfComponentInstantiationFailed</fmAlarmTypeId>
      <majorType>18568</majorType>
      <minorType>131074</minorType>
      <moClasses>Amf</moClasses>
      <specificProblem>COM SA, AMF Component Instantiation Failed =>
      </specificProblem>
      <eventType>PROCESSINGERRORALARM</eventType>
      <probableCause>418</probableCause>
      <isStateful>true</isStateful>
      <additionalText></additionalText>
    </FmAlarmType>
  </FmAlarmModel>
  <FmAlarmModel>
    <fmAlarmModelId>CertM</fmAlarmModelId>
    <FmAlarmType>
      <fmAlarmTypeId>ComSaProxyStatusOfAComponentChangedToUnproxied =>
      </fmAlarmTypeId>
      <majorType>18568</majorType>
      <minorType>131078</minorType>
      <moClasses>Amf</moClasses>
      <specificProblem>COM SA, Proxy Status of a Component Changed to =>
      Unproxied</specificProblem>
      <eventType>PROCESSINGERRORALARM</eventType>
      <probableCause>418</probableCause>
      <isStateful>true</isStateful>
      <additionalText></additionalText>
    </FmAlarmType>
  </FmAlarmModel>
  <FmAlarmModel>
    <fmAlarmModelId>CertM</fmAlarmModelId>
    <FmAlarmType>
      <fmAlarmTypeId>CertMCertificateToExpire</fmAlarmTypeId>
      <majorType>193</majorType>
      <minorType>6946818</minorType>
      <moClasses>NodeCredential</moClasses>
      <specificProblem>Certificate Management, the Certificate is to =>
      Expire</specificProblem>
      <eventType>PROCESSINGERRORALARM</eventType>
      <probableCause>351</probableCause>
      <isStateful>true</isStateful>
      <additionalText>The threshold before certificate expiration has =>
      been crossed, and the certificate should be renewed to prevent a secure service failure. =>
      </additionalText>
    </FmAlarmType>
  </FmAlarmModel>
  <FmAlarmModel>
    <fmAlarmModelId>CertM</fmAlarmModelId>
    <FmAlarmType>
      <fmAlarmTypeId>CertMCertificateNotAvailable</fmAlarmTypeId>
      <majorType>193</majorType>
      <minorType>6946817</minorType>
      <moClasses>NodeCredential</moClasses>
      <specificProblem>Certificate Management, a Valid Certificate is Not =>
      Available</specificProblem>
      <eventType>OPERATIONALVIOLATION</eventType>
      <probableCause>414</probableCause>
      <isStateful>true</isStateful>
      <additionalText>The certificate provided by the alarming object is =>
      expired, revoked, or unavailable, which may result in the failure of a secure service using this =>
      certificate.</additionalText>
    </FmAlarmType>
  </FmAlarmModel>
  <FmAlarmModel>
    <fmAlarmModelId>CertM</fmAlarmModelId>
    <FmAlarmType>
      <fmAlarmTypeId>CertMAutomaticEnrollmentFailed</fmAlarmTypeId>
      <majorType>193</majorType>
      <minorType>6946819</minorType>
      <moClasses>NodeCredential</moClasses>
      <specificProblem>Certificate Management, Automatic Enrollment =>
      Failed</specificProblem>
      <eventType>PROCESSINGERRORALARM</eventType>
      <probableCause>307</probableCause>
      <isStateful>true</isStateful>
      <additionalText>Automatic enrollment or renewal failed to execute =>
      due to local misconfiguration or remote enrollment service denial.</additionalText>
    </FmAlarmType>
  </FmAlarmModel>

```

```

        <FmAlarmModel>
          <fmAlarmModelId>LocalAuthenticationMethod</fmAlarmModelId>
          <FmAlarmType>
            <fmAlarmTypeId>LocalAuthNAAuthenticationFailureLimitReached =>
            </fmAlarmTypeId>
            <majorType>193</majorType>
            <minorType>6946820</minorType>
            <moClasses>AdministratorAccount</moClasses>
            <specificProblem>Local Authentication, Authentication Failure Limit =>
            Reached</specificProblem>
            <eventType>SECURITYSERVICEORMECHANISMSVIOLATION</eventType>
            <probableCause>401</probableCause>
            <isStateful>true</isStateful>
            <additionalText>The authentication failure limit is reached based =>
on the configured threshold. A password attack is suspected that should be isolated from the ME. =>
            </additionalText>
          </FmAlarmType>
        </FmAlarmModel>
      </Fm>
    </SystemFunctions>
  </ManagedElement>
</data>
</rpc-reply>]]>

```

<ManagedElement> is the root of the MO tree, that is, the whole MO subtree contained in the datastore. It is returned if parameter <filter> contains <ManagedElement/> or <ManagedElement></ManagedElement>, see Example 32.

```

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

```

**Example 32** Filtered Operation <get> on ManagedElement

#### 5.1.2.4

#### 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 33 and Example 34.

```

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

```

**Example 33** Operation <get> Request with DN Filter on MO Fm



```
<?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>
      <SystemFunctions>
        <systemFunctionsId>1</systemFunctionsId>
        <Fm xmlns="urn:com:ericsson:ecim:ComFm">
          <fmId>1</fmId>
          <FmAlarmModel>
            <fmAlarmModelId>LOTC</fmAlarmModelId>
            <FmAlarmType>
              <fmAlarmTypeId>DiskReplicationCommunication</fmAlarmTypeId>
              <majorType>193</majorType>
              <minorType>3341942788</minorType>
              <moClasses>SafNode</moClasses>
              <specificProblem>LOTC Disk Replication Communication</specificProblem>
              <eventType>ENVIRONMENTALALARM</eventType>
              <probableCause>418</probableCause>
              <isStateful>true</isStateful>
              <additionalText>dummy</additionalText>
            </FmAlarmType>
            <FmAlarmType>
              <fmAlarmTypeId>DiskReplicationConsistency</fmAlarmTypeId>
              <majorType>193</majorType>
              <minorType>3341942790</minorType>
              <moClasses>SafNode</moClasses>
              <specificProblem>LOTC Disk Replication Consistency</specificProblem>
              <eventType>ENVIRONMENTALALARM</eventType>
              <probableCause>418</probableCause>
              <isStateful>true</isStateful>
              <additionalText>dummy</additionalText>
            </FmAlarmType>
            <FmAlarmType>
              <fmAlarmTypeId>DiskUsage</fmAlarmTypeId>
              <majorType>193</majorType>
              <minorType>3341942787</minorType>
              <moClasses>SafNode</moClasses>
              <specificProblem>LOTC Disk Usage</specificProblem>
              <eventType>ENVIRONMENTALALARM</eventType>
              <probableCause>418</probableCause>
              <isStateful>true</isStateful>
              <additionalText>dummy</additionalText>
            </FmAlarmType>
            <FmAlarmType>
              <fmAlarmTypeId>EthernetBonding</fmAlarmTypeId>
              <majorType>193</majorType>
              <minorType>3341942786</minorType>
              <moClasses>SafNode</moClasses>
              <specificProblem>LOTC Ethernet Bonding</specificProblem>
              <eventType>ENVIRONMENTALALARM</eventType>
              <probableCause>418</probableCause>
              <isStateful>true</isStateful>
              <additionalText>dummy</additionalText>
            </FmAlarmType>
          </Fm>
        </SystemFunctions>
      </ManagedElement>
    </data>
  </rpc-reply>

```

**Example 34** Operation <get> Reply Message with MO Fm

```

    <FmAlarmType>
      <fmAlarmTypeId>MemoryUsage</fmAlarmTypeId>
      <majorType>193</majorType>
      <minorType>3341942789</minorType>
      <moClasses>SafNode</moClasses>
      <specificProblem>LOTC Memory Usage</specificProblem>
      <eventType>ENVIRONMENTALALARM</eventType>
      <probableCause>418</probableCause>
      <isStateful>true</isStateful>
      <additionalText>dummy</additionalText>
    </FmAlarmType>
    <FmAlarmType>
      <fmAlarmTypeId>TimeSynchronization</fmAlarmTypeId>
      <majorType>193</majorType>
      <minorType>3341942785</minorType>
      <moClasses>SafNode</moClasses>
      <specificProblem>LOTC Time Synchronization</specificProblem>
      <eventType>ENVIRONMENTALALARM</eventType>
      <probableCause>418</probableCause>
      <isStateful>true</isStateful>
      <additionalText>dummy</additionalText>
    </FmAlarmType>
  </FmAlarmModel>
</Fm>
</SystemFunctions>
</ManagedElement>
</data>
</rpc-reply>
]]>]]>

```

### 5.1.2.5 DN Filter on Multiple Branches

The DN filter can contain multiple combined DN elements by either of the ways shown in Example 35 and Example 36.

```

<?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>NODE06ST</managedElementId>
        <NCTest>
          <nCTestId>1</nCTestId>
          <XThing>
            <xThingId>1</xThingId>
          </XThing>
          <YThing>
            <yThingId>1</yThingId>
          </YThing>
        </NCTest>
      </ManagedElement>
    </filter>
  </get>
</rpc>]]>]]>

```

**Example 35** Operation <get> 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>NODE06ST</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>NODE06ST</managedElementId>
        <NCTest>
          <nCTestId>1</nCTestId>
          <YThing>
            <yThingId>1</yThingId>
          </YThing>
        </NCTest>
      </ManagedElement>
    </filter>
  </get-config>
</rpc>]]>]]>
```

**Example 36** Operation `<get-config>` Request with DN Filter on Multiple MOs

The operation reply contains both requested elements, see Example 37.

```
<?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>NODE06ST</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 37** Multiple Requested Elements in Operation Response

### 5.1.2.6 Filtered Operation `<get>` on Single-Valued Attribute

Selected attributes are returned if filter `<get>` contains the XML representation of the attributes, see Example 38 and Example 39.

```
<?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>NODE06ST</managedElementId>
        <timeZone></timeZone>
        <siteLocation></siteLocation>
      </ManagedElement>
    </filter>
  </get-config>
</rpc>]]>]]>
```

### Example 38 Operation <get> 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>NODE06ST</managedElementId>
      <siteLocation>siteLocationValue</siteLocation>
      <timeZone>timeZoneValue</timeZone>
    </ManagedElement>
  </data>
</rpc-reply>
]]>]]>
```

### Example 39 Operation <get> Reply with MO Attributes

The reply message in Example 40 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>NODE06ST</managedElementId>
      <userLabel unset="true"></userLabel>
    </ManagedElement>
  </data>
</rpc-reply>
]]>]]>
```

### Example 40 Operation <get> Reply with Attribute Value Not Set

The error message in Example 41 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="100">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>unknown-attribute</error-tag>
    <error-severity>error</error-severity>
    <error-info>
      <bad-attribute>userLabel1</bad-attribute>
      <bad-element>ManagedElement</bad-element>
    </error-info>
    <error-message xml:lang="en">Failed to add attribute</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

### Example 41 Operation <get> Reply with Attribute Not Defined



### 5.1.2.7

### Filtered Operation <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 42 and Example 43. Filter on a struct member attribute (such as anInt64Attr in Example 43) 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>NODE06ST</ex:managedElementId>
        <NCTest>
          <nCTestId>1</nCTestId>
          <SimpleStruct>
            <simpleStructId>1</simpleStructId>
            <attrFileData/>
          </SimpleStruct>
        </NCTest>
      </ManagedElement>
    </filter>
  </get>
</rpc>]]>]]>
```

#### Example 42 Operation <get> 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>NODE06ST</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 43 Operation <get> Reply with MO Attributes

### 5.1.3

### Request Event Stream Discovery

As a function defined in RFC 5277, a NETCONF client can retrieve the list of supported event streams from a NETCONF server by operation <get> on element <streams>, see Example 44. Only the mandatory stream NETCONF is supported by the Ericsson NETCONF interface. 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 44 Request Event Stream Discovery

As shown in Example 45, 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 45 Event Stream Discovery Reply

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

## 5.1.4

### Support for Nodes

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



#### 5.1.4.1 Containment Node

A containment node is a node containing child elements within a subtree filter. 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.

Element `<ManagedElement>` is a containment node in Example 46.

```
<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>NODE06ST</managedElementId>
        <userLabel/>
      </ManagedElement>
    </filter>
  </get>
</rpc>]]>]]>
```

#### Example 46 Request 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>NODE06ST</managedElementId>
      <userLabel>UserLabelValue</userLabel>
    </ManagedElement>
  </data>
</rpc-reply>
]]>]]>
```

#### Example 47 Containment Node Reply

#### 5.1.4.2 Selection Node

A selection node is an empty leaf node within a filter. The selection node represents an explicit selection filter on the underlying data model. Presence of any selection nodes within a set of sibling nodes causes 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, such as `<foo/>`, or with explicit start and end tags, such as `<foo> </foo>`. Any white-space characters are ignored in this form.

In Example 48, the request `ManagedElement` is a containment node with content match, `NCTest` is a containment node, and `Xthing` is an empty leaf node 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 are not returned, except for attribute `id` that is always returned. Attributes specified in a content match are always returned.

In Example 48, `managedElementId` is returned but no other attributes of `ManagedElement`. If a containment node only contains content match nodes, all attributes and children to that node are returned, which is the trivial case already implemented for the attribute `id` match.

The filter is a request for all `XThing` instances under all `NCTest` instances under `ManagedElement` instances, where `managedElementId=NODE06ST`. This happens to be the value of DN attribute `id` for this MO. If more attributes are included under `ManagedElement` in the filter, they are evaluated as a logical **AND** that must be true for a filter match, as in Example 49. 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>NODE06ST</managedElementId>
        <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
          <XThing/>
        </NCTest>
      </ManagedElement>
    </filter>
  </get>
</rpc>]]>]]>
```

#### Example 48 Request 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>NODE06ST</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 49 Selection Node Reply

### 5.1.4.3

#### Content Match Node

A content match node is a leaf node containing simple content. It represents an exact match filter on the leaf node 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 a logical **AND** expression.
- Filtering of mixed content is not supported.



- Filtering of list content is not supported.
- Filtering of white-space-only content is not supported.
- A content match node must contain non-white-space characters. An empty element, such as `<foo></foo>`, is interpreted as a selection node, such as `<foo/>`.
- Leading and trailing white-space characters are ignored. However, any white-space 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, the filter output nodes are selected as follows:

- Each content match node in the sibling set is included in the filter output.
- If any containment nodes are present in the sibling set, they are processed further and included if any nested filter criteria is also met.
- If any selection node is present in the sibling set, all 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, 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 sibling content match node test is false, no further filter processing is performed on that sibling set, and no sibling subtrees are selected by the filter, including the content match nodes.

Example 50 shows a request for subtree `XThing`, where `xthingId=1` and `rwAttr2=1`, under all `NCTest` instances under `ManagedElement`, where `managedElementId=NODE06ST`. As no selection node is present in `XThing`, all its attributes and subtrees are returned in Example 51.

```
<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>NODE06ST</managedElementId>
        <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
          <XThing>
            <xthingId>1</xthingId>
            <rwattr1>RW-One</rwattr1>
          </XThing>
        </NCTest>
      </ManagedElement>
    </filter>
  </get>
</rpc>]]>]]>
```

**Example 50** Request 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>NODE06ST</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 Content Match Node Reply*

## Error during Content Match Node Processing

If the content match specified in the subtree filtering is false, that it, does not match with any of the subtree filtering, an error message is returned, see Example 52.

```
<?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=NODE06ST,NCTest,⇒
XThing=1 is not available with requested content (no instance).⇒
  </error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

*Example 52 Error Message during Content Match Node Processing*

### 5.1.5

## Optional MO Instances Specification in <get> and <get-config> Requests

Operations <get> and <get-config> are supported without specifying the MO instances in the 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 53 Optional MO Instances 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 xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>NODE06ST</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 54 Optional MO Instances Reply

**Note:** For a few types of `<get>` and `<get-config>` 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 in the system during the execution of the request.

For the `<get>` request in Example 55, 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>NODE06ST</managedElementId>
        <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
          <nCTestId>1</nCTestId>
          <XThing/>
        </NCTest>
      </ManagedElement>
    </filter>
  </get>
</rpc>]]>]]>
```

### Example 55 Operation `<get>` Request

### 5.1.6 Operation <get> with RFC Compliance Latest Mode

Operation <get> does not return any error if the requested MO does not exist.

If operation <get> contains request for non-instantiated child MOC as shown in Example 56, the response is generated with empty <data> tags as shown in Example 57. Here xThingId = 2 and yThingId = 2 are not instantiated, and 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 56** 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 57** Response with Empty <data> Tags

If operation <get> requests non-instantiated MOC as shown in the Example 58, the response to operation <get> displays empty <data> tags as shown in Example 57.

```
<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 58** Request for Non-Instantiated MOC

If the <get> request contains multiple MOC requests where one MOC exists and the other MOC does not exist, only the existing MOC is displayed in the response to the <get> request as shown in Example 59 and Example 60.



```
<?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 59 Request Containing Instantiated and Non-Instantiated MOCs

```
<?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 60 Response Containing Instantiated MOC

## 5.2 Operation <get-config>

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 extra parameter, <source>, identifying the configuration datastore being queried. The source parameter must be running, see Example 61.
- The reply message of operation <get-config> contains configuration data but no state data. For definition of configuration and state data, see Section 4.3 Configuration and State Data on page 16.

```
<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>NODE06ST</managedElementId>
        <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
          <nCTestId>1</nCTestId>
          <XThing>
            <xThingId>1</xThingId>
          </XThing>
        </NCTest>
      </ManagedElement>
    </filter>
  </get-config>
</rpc>]]>]]>
```

*Example 61 Operation <get-config> Request Message*

## 5.3 Operation <edit-config>

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

### Parameters

Operation <edit-config> can be requested with the following parameters:

- **target** – Must be running, see Example 62.
- **default-operation** – Optional. If provided, it must have one of the following values:
  - **merge** – The configuration data in parameter <config> is merged with the configuration at the corresponding level in target datastore, see Example 62.
  - **replace** – The configuration data in parameter <config> replaces the configuration in target datastore.
  - **none** – Target datastore is unaffected by the configuration in parameter <config>, unless and until the incoming configuration data uses attribute **operation** to request a different operation.
- **error-option** – 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. It indicates that this <error-option> is not supported and the session is closed if the session commit behavior is “commit at <close-session>”. Otherwise the session continues.
- **config** – Mandatory. Identifies the target configuration, consisting of one or more MOs and attributes. That is, multiple MOs or MO subtrees can be changed with one <edit-config> operation.



Elements in subtree `<config>` can contain operation attributes identifying the points in the configuration where the operation is to be performed. Attribute `operation` can appear in multiple elements throughout subtree `<config>`. Attribute `operation` 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 parameter `<target>`. This is the default behavior. For examples, see Section 5.3.5 Operation `<edit-config>` with Option `merge` on page 53.
- `create` – The configuration data identified by the element containing this attribute is added to the configuration only if the configuration data does not exist in the configuration datastore. If the configuration data exists, element `<rpc-error>` is returned with `<error-tag>` value `data-exists`. For examples, see Section 5.3.2 Operation `<edit-config>` with Option `create` on page 49.
- `delete` – The configuration data identified by the element containing this attribute is deleted from the configuration only if the configuration data exists in the configuration datastore. If the configuration data does not exist, element `<rpc-error>` is returned with `<error-tag>` value `data-missing`. For examples, see Section 5.3.3 Operation `<edit-config>` with Option `delete` on page 51.
- `replace` – The configuration data identified by the element containing this attribute replaces any related configuration in the configuration datastore identified by parameter `<target>`. If no such configuration data exists in the configuration datastore, it is created. Unlike operation `<copy-config>`, which replaces the entire target configuration, only the configuration present in parameter `<config>` is affected. For examples, see Section 5.3.6 Operation `<edit-config>` with Option `replace` on page 59.
- `remove` – The configuration data identified by the element containing this attribute is deleted from the configuration if the configuration data exists in the configuration datastore. If the configuration data does not exist, operation `remove` is silently ignored by the server. This option is not supported, as a limitation to RFC 6241.

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

**Note:** Parameter `<test-option>` must not be present, as capability `:validate` is not supported.

```
<?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>NODE06ST</managedElementId>
        <SystemFunctions>
          <systemFunctionsId>1</systemFunctionsId>
          <SysM>
            <sysMId>1</sysMId>
            <NtpServer>
              <ntpServerId>1</ntpServerId>
              <serverAddress>127.0.0.1</serverAddress>
            </NtpServer>
          </SysM>
        </SystemFunctions>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>]]>>
```

*Example 62 Default Operation Tag with Option merge*

## Positive Response

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

The commit behavior is as follows:

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

## Negative Response

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

- Validation rules derived from the model properties are violated, as described in Section 7 on page 83.
- An implementation-specific validation rule defined for the MO instance or the MO attribute is violated.
- The MO attribute, the MO instance, or its parent MO is locked. For details on locking, see Section 9.2 Locking on page 107.
- An internal processing error occurred, as described in Section 3.3 Error Messages on page 11.



All the changes issued in the failing `<edit-config>` request are rolled back. The effect on 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 response `<rpc-error>`, the NETCONF server closes the session. Changes done in the same session by previous successful `<edit-config>` operations are also rolled back.
- 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 details on transaction commit, see Section 9.3 Transaction Commit on page 108.

### 5.3.1 Error with Target Set to Start Up Datastore

Operation `<edit-config>` loads or changes all or part of a specified configuration in the running datastore. If the startup datastore is set as the target datastore, the RPC error message in Example 63 is returned.

```
<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 63 Error Message When Startup Datastore is Used*

### 5.3.2 Operation `<edit-config>` with Option `create`

Operation `create` results in MO instance creation but creation of attributes is not supported. Use option `merge` to assign values to an unset attribute for an existing MO.

#### 5.3.2.1 Create MO Instances

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

An example of creating two *NtpServer* MOs is shown in Example 64.

```
<?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>NODE06ST</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 64** Create MO Instances

### 5.3.2.2

#### Operation create with RFC Compliance Latest Mode

Operation `<edit-config>` with option `create` provided at the parent MOC is inherited to the child MOCs, if any. This applies also if `<default-operation>` contains option `none`.

Operation `<edit-config>`, as shown in Example 65, is a request to create an MO. Here along with creation of parent MOC `AThing`, also `AAThing` child MOC is created, also 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 65** Request for Operation create with Default-Operation Set to None





### 5.3.3 Operation <edit-config> with Option delete

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

#### 5.3.3.1 Delete MO Instances

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

An example of an `NtpServer` deletion request is shown in Example 66.

```
<?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>NODE06ST</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 66 Delete MO Operation Request*

#### 5.3.3.2 Delete MO Attribute Value

Operation <edit-config> triggers an attribute value deletion request if attribute `xc:operation="delete"` 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>NODE06ST</managedElementId>
        <userLabel xc:operation="delete"/>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>]]>]]>
```

*Example 67 Delete MO Attribute Operation Request*

### 5.3.3.3 Delete Sequence Element

If a sequence element (multi-value) contains attribute `xc:operation="delete"` in operation `<edit-config>`, 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 are to delete all occurrences of the specified value.

A request to delete a sequence element is shown in Example 68.

```
<?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>
        <exampleStructAttribute struct="FileDataMv">
          <exampleStructAttributeId>1</exampleStructAttributeId>
          <attrFileData>
            <int64Member xmlns:erince="urn:com:ericsson:netconf:operation:1.0" =>
xc:operation="delete" erince:position="all">44</int64Member>
          </attrFileData>
        </exampleStructAttribute>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>]]>]]>
```

*Example 68 Delete Individual Sequence Element Operation Request*

### 5.3.3.4 Delete Struct

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

### 5.3.3.5 Delete Struct Member

The deletion of individual sequence element in a struct member is also possible (see Section 5.3.3.3 Delete Sequence Element on page 52), with the 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, see Example 69.



```
<?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>
</xml>
```

**Example 69** *Error Message for Deleting Sequence Element in Non-Singleton Struct*

### 5.3.4 Operation <edit-config> with Default Option

If no attribute `operation` is present, the operation is performed according to option `merge`, as described in Section 5.3.5 Operation <edit-config> with Option merge on page 53.

### 5.3.5 Operation <edit-config> with Option merge

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

Operation `merge` triggers the following:

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

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

#### 5.3.5.1 Change Single-Valued Attribute

The operation in Example 70 requests to change attribute `administrative State`.

```
<?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>NODE06ST</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">
                <localAuthorizationMethodId>1</localAuthorizationMethodId>
                <administrativeState>UNLOCKED</administrativeState>
              </LocalAuthorizationMethod>
            </UserManagement>
          </SecM>
        </SystemFunctions>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>]]>]]>
```

**Example 70** *Change Attribute administrativeState*

### 5.3.5.2 Change Sequence Attribute

The operation in Example 71 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 operation `merge`.

```
<?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>NODE06ST</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 71** *Change Sequence Attribute*

Merge of a sequence attribute is supported, but not merge of sequence elements.



### 5.3.5.3 Change Sequence of Struct

The attributes defined as a sequence of struct can be changed, see Example 72. 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 operation merge.

```
<?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>NODE06ST</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 72 Change Sequence of Struct*

### 5.3.5.4 Create MO with Single-Valued Attributes

The operation in Example 73 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>NODE06ST</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 73 Create MO with Single-Valued Attributes*

### 5.3.5.5

#### Create MO with Structure Data Type

The operation in Example 74 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>NODE06ST</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 74 Create MO with Structure Data Type*



### 5.3.5.6

#### Create MO with Exclusive Structure Data Type

The operation in Example 75 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 flag `isExclusive`, meaning that only one member of the structure can be present in the structure.

Attribute `attrSimpleStruct` is defined as an exclusive structure of structure members `hostId` and `ipAddress`. Example 75 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>NODE06ST</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 75 Create MO with Exclusive Structure Data Type

The configuration in Example 76 is invalid, 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>NODE06ST</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 76 Violation of Exclusivity

An error message indicating violation of exclusivity is shown in Example 77.



```
<?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 77 Error Message for Violation of Exclusivity*

### 5.3.5.7 Set ECIM Password Attribute

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

- If the ECIM password type attribute is configured with `cleartext` and with password attribute value in the NETCONF request (see Example 78), the password attribute value is encrypted and the encrypted password value is stored.
- If the ECIM password type attribute is configured without `cleartext` and with password attribute value in the NETCONF request (see Example 79), the password attribute value is stored as it is (that is, whatever 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 78 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 79 Request for Setting an ECIM Password without <cleartext/> Tag*

### 5.3.6 Operation <edit-config> with Option replace

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

#### 5.3.6.1 Replace MO Instance

The operation in Example 80 requests to replace 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>NODE06ST</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 80 Replace with Single-Valued Attribute*

### 5.3.6.2 Replace MO Attribute

Replacing an MO attribute is not supported.

### 5.3.6.3 Operation replace with RFC Compliance Latest Mode

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

The operation shown in Example 81 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, 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 81 Request for <edit-config> Operation replace*



Operation `replace` is not allowed on system-created MOs and result in an error as shown in Example 82 and Example 83.

```
<?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 82** Request for Operation `<edit-config>` 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>
</rpc>]]>]]>
```

**Example 83** Error Message for Replacing a System-Created MO

### 5.3.7

### Whitespace, Tab, and New Line Handling in `<edit-config>`

When the `<edit-config>` request for an integer or float datatype along with leading and trailing spaces is given as shown in the Example 84, then an error message is thrown as shown in the Example 85.

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
          <nCTestId>1</nCTestId>
          <BasicThing>
            <basicThingId>1</basicThingId>
            <anInt16Attr> 8 </anInt16Attr>
          </BasicThing>
        </NCTest>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>
</rpc>]]>]]>
```

**Example 84** `<edit-config>` Operation for an Integer Attribute Change

```
<?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>anInt16Attr</bad-attribute>
      <bad-element>BasicThing</bad-element>
    </error-info>
    <error-message xml:lang="en">Invalid value 8 for an
      integer type attribute anInt16Attr</error-message>
    </rpc-error>
  </rpc-reply>
</></>
```

**Example 85** *Error Message for <edit-config> Operation when an Integer Value is Given with Leading and Trailing Spaces*

**Note:** NETCONF takes the value as it is without trimming the leading and trailing spaces for int/float values.

When <edit-config> request for boolean value with upper/mixed case is given as shown in the Example 86, then error message is thrown as shown in the Example 87.

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
          <nCTestId>1</nCTestId>
          <BasicThing>
            <basicThingId>1</basicThingId>
            <aBoolAttr>TRUE</aBoolAttr>
          </BasicThing>
        </NCTest>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>
</></>
```

**Example 86** *<edit-config> Operation for an Uppercase Boolean Value*

```
<?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>aBoolAttr</bad-attribute>
      <bad-element>BasicThing</bad-element>
    </error-info>
    <error-message xml:lang="en">
      Invalid value TRUE for attribute aBoolAttr
    </error-message>
  </rpc-error>
</rpc-reply>
</></>
```

**Example 87** *Error Message for <edit-config> Operation with an Uppercase Boolean Value*



**Note:** Only lowercase boolean value is accepted in NETCONF.

When the `<edit-config>` request for a string datatype along with leading and trailing spaces is given as shown in the Example 88, then the value is taken as it is without any change as shown in Example 89.

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
          <nCTestId>1</nCTestId>
          <BasicThing>
            <basicThingId>1</basicThingId>
            <aStringAttr COM />
          </BasicThing>
        </NCTest>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>
]]>]]>
```

#### Example 88 `<edit-config>` Request for String Type

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
  <data>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
        <nCTestId>1</nCTestId>
        <BasicThing>
          <aStringAttr COM />
        </BasicThing>
      </NCTest>
    </ManagedElement>
  </data>
</rpc-reply>
]]>]]>
```

#### Example 89 `<edit-config>` Reply Without Removing the Spaces

When the `<edit-config>` request only with spaces is given as shown in Example 90, then the value is taken by removing the spaces as shown in Example 91.



```
<?xml version="1.0" encoding="UTF-8"?>
<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>
          <nCTestId>1</nCTestId>
          <BasicThing>
            <basicThingId>1</basicThingId>
            <aStringAttr>          </aStringAttr>
          </BasicThing>
        </NCTest>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>
]]>]]>
```

**Example 90** *<edit-config> Request with Spaces*

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
  <data>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
        <nCTestId>1</nCTestId>
        <BasicThing>
          <aStringAttr></aStringAttr>
        </BasicThing>
      </NCTest>
    </ManagedElement>
  </data>
</rpc-reply>
]]>]]>
```

**Example 91** *<edit-config> Reply by Removing the Spaces*

**Note:** Multiple spaces/single space is not possible to provide with NETCONF. As of now NETCONF takes as an empty value.

When the <edit-config> request with new line is given as shown in Example 92, then the value is taken by removing the spaces as shown in Example 93.



```
<?xml version="1.0" encoding="UTF-8"?>
<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>
          <nCTestId>1</nCTestId>
          <BasicThing>
            <basicThingId>1</basicThingId>
            <aStringAttr>This is Com
              COM is used in CBA and
              Non-CBA
            </aStringAttr>
          </BasicThing>
        </NCTest>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>
]]>]]>
```

#### Example 92 <edit-config> Request with New Lines

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
  <data>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <NCTest xmlns="urn:com:ericsson:ecim:ncmom1">
        <nCTestId>1</nCTestId>
        <BasicThing>
          <aStringAttr>This is Com COM is used in CBA and Non-CBA
        </aStringAttr>
        </BasicThing>
      </NCTest>
    </ManagedElement>
  </data>
</rpc-reply>
]]>]]>
```

#### Example 93 <edit-config> Reply by Ignoring the New Lines

**Note:** New lines are ignored for string data type.

## 5.4 Operation <action>

Action execution can be requested by operation <action>, see Section 10 on page 111.

## 5.5 Operation <create-subscription>

This section describes the RFC 5277-defined operation <create-subscription> and the Ericsson NETCONF-specific filter extension, defined in Section 11.5.1 Operation <create-subscription> on page 117.

This operation initiates an event notification subscription sending asynchronous event notifications to the initiator of the command in the NETCONF session until the subscription terminates.

Although the session has an active notification subscription, operations are processed, as RFC 5277-defined interleave capability is supported. Only one subscription per session is supported.

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

## Parameters

The parameters are as follows:

- `<stream>` – Optional parameter indicating 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, as described in Section 5.1.3 Request Event Stream Discovery on page 37.
- `<filter>` – Optional parameter indicating which subset of all possible events is of interest. If not present, all events not precluded by other parameters are sent. The parameter format 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 Ericsson `NETCONF` proprietary elements, as described in Section 11.5.1 Operation `<create-subscription>` on page 117:

- `<event>`
- `<filterType>`
- `<filterValue>`

Both the standard compliant and the deviation are supported. However, both cannot be present in the same `<create-subscription>` message.

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

For 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 configuration management events. If the system is configured for internally generating the events, only one notification for the root of the deleted MO tree is generated. If the system is configured for receiving the configuration management events from the middleware through the support agent, notifications for the other MOs in the subtree can also be produced, depending on how the middleware and the support agent are implemented.





As 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 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 triggering the replay feature and indicates that the replay is to start at the time specified. If `<startTime>` is not present, it is not a replay subscription. `<startTime>` must not be later than the current time. If `<startTime>` is earlier than the log can support, notification `<replayTimeUnsupported>` (see Section 11.6.2 Notification `<replayTimeUnsupported>` on page 119) 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.

The date and time of the earliest logged notification is available in parameter `replayLogCreationTime` of the event stream and can be retrieved, as described in Section 5.1.3 Request Event Stream Discovery on page 37.

- `<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 terminates. `<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.

**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 2.3 Session Inactivity Timer on page 5.

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>` has format `+/-hh' : 'mm` or `Z` denoting an offset of 00:00.

Example: `2011-11-03T13:54:26+02:00`.

- `<aggregationLevel>` – Optional parameter when user sends aggregated notifications over the NBI.

User can set the number of events to be aggregated under one notification as per specified value, as shown in Example 99.

Acceptable options:

- `all` – When specified under `create-subscription`, all the events under one transaction are aggregated in one notification.
- Integers in the range [1,65535] – All events of transaction are displayed as per specified value under single notification and remaining notifications that were not yet sent in the transaction are sent in a new `<notification>` tag.

If `<aggregationLevel>` tag is not specified, number of events are considered as 1, by default and MO level notifications are displayed.

If some value other than a positive integer value or `all` is specified, an error is thrown as shown in Example 100.

If the client does not specify the new capability and `aggregationLevel` is specified in `create-subscription`, the existing error-message shown in Example 101 is continued to maintain BC.

### Positive Response

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

### Negative Response

If the request cannot be completed, element `<rpc-error>` is included in `<rpc-reply>`. A subscription request fails if a filter with invalid syntax is provided or if a name of a non-existent stream is provided.

### Examples

The operation in Example 94 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 94 Create Subscription without Filter*

A subscription with the filter in Example 95 receives notifications upon creation or deletion of any MO starting with DN `ManagedElement=NODE06ST, 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=NODE06ST,SystemFunctions=1,Snmp.*
        </filterValue>
      </event>
      <event>
        <filterType>objectDeleted</filterType>
        <filterValue>ManagedElement=NODE06ST,SystemFunctions=1,Snmp.*
        </filterValue>
      </event>
      <event>
        <filterType>attributeChanged</filterType>
        <filterValue>.*</filterValue>
      </event>
    </filter>
  </create-subscription>
</rpc>]]>]]>
```

### **Example 95**    *Create Subscription with 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">
    <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 96**    *Create 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">
      <ManagedElement>
        <managedElementId>NODE06ST</managedElementId>
        <TestRootMoc>
          <YThing>
            <rwattr2>2</rwattr2>
          </YThing>
        </TestRootMoc>
      </ManagedElement>
      <ManagedElement>
        <managedElementId>NODE06ST</managedElementId>
        <TestRootMoc>
          <YThing/>
        </TestRootMoc>
      </ManagedElement>
    </filter>
  </create-subscription>
</rpc>]]>]]>
```

### **Example 97**    *Create Subscription with Content Match and MO Selection (Standard Subtree Filters)*



```
<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>NODE06ST</managedElementId>
        <TestRootMoc xmlns="urn:com:ericsson:ecim:com_test_cli_1">
          <testRootMocId>1</testRootMocId>
        </TestRootMoc>
      </ManagedElement>
      <ManagedElement>
        <managedElementId>NODE06ST</managedElementId>
        <TestRootMoc>
          <YThing>
            <rwattr2/>
          </YThing>
        </TestRootMoc>
      </ManagedElement>
    </filter>
  </create-subscription>
</rpc>]]>]]>
```

**Example 98** *Create Subscription with Key Attribute Match and Attribute Selection (Standard Subtree Filters)*

```
<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">
    <aggregationLevel>1</aggregationLevel>
  </create-subscription>
</rpc>
]]>]]>
```

**Example 99** *Creating Subscription with aggregationLevel*

The operation in Example 99 requests to create a subscription with aggregationLevel 1.

```
?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>bad-element</error-tag>
    <error-severity>error</error-severity>
    <error-info>
      <bad-element>aggregationLevel</bad-element>
    </error-info>
    <error-message xml:lang="en">0 is not acceptable value for aggregationLevel.
      Acceptable values are integers from the interval [1,65535] or 'all'.</error-message>
  </rpc-error>
</rpc-reply>
```

**Example 100** *Error Message Displayed for Invalid Values in aggregationLevel*

```
<?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>rpc</error-type>
    <error-tag>unknown-element</error-tag>
    <error-severity>error</error-severity>
    <error-info>
      <bad-element>aggregationLevel</bad-element>
    </error-info>
    <error-message xml:lang="en">An unexpected element is present.</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

**Example 101** *Error Message Displayed without aggregationLevel Capability*



### 5.5.1 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 terminates.

If `<startTime>` is earlier than the log can support, the replay begins with `ReplayStartTimeUnsupported` followed by the earliest available notifications, see Example 102.

```
<?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=NODE06ST">
      <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=NODE06ST">
      <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=NODE06ST">
      <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 102 Start Time Earlier Than Log Can Support*

#### 5.5.1.1 Notification Replay with aggregation:1.0 Capability

When the client specifies the aggregation:1.0 capability in the client `hello` and `aggregationLevel` in the create-subscription request, then notification

replay is displayed as per the aggregationLevel (see also Section 11.6.4 MO Change Notifications with aggregation:1.0 Capability on page 124).

Suppose that there are 5 available events of one transaction and aggregationLevel is 2, then 3 replay notifications (2 events, 2 events, 1 event) are displayed, followed by replayComplete notification.

In Example 103, aggregationLevel=all is chosen, therefore all the available events of a transaction are displayed in a single notification tag followed by replayComplete notification as shown in Example 104.

```
<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">
    <startTime>2016-09-26T10:40:00+01:00</startTime>
    <aggregationLevel>all</aggregationLevel>
  </create-subscription>
</rpc>
]]>]]>
```

### Example 103 Create Subscription with <aggregationLevel> and <startTime> Tags

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-26T09:38:00+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <objectCreated dn="ManagedElement=1,NCTest=1,XThing=2">
      <attr name="rwattr1">
        <v>RW-One</v>
      </attr>
      <attr name="spacestrattr">
        <v>"abc"</v>
      </attr>
      <attr name="rwattr2">
        <v>2</v>
      </attr>
    </objectCreated>
    <objectDeleted dn="ManagedElement=1,NCTest=1,XThing=3"/>
    <AVC dn="ManagedElement=1,NCTest=1,XThing=2">
      <attr name="rwattr1">
        <v>1</v>
      </attr>
      <attr name="rwattr2">
        <v>2</v>
      </attr>
    </AVC>
  </events>
</notification>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-26T11:31:27+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <objectDeleted dn="ManagedElement=1,NCTest=1,XThing=2"/>
  </events>
</notification>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-26T11:53:59+02:00</eventTime>
  <replayComplete xmlns="urn:ietf:params:xml:ns:netmod:notification"/>
</notification>
]]>]]>
```

### Example 104 Notification for Aggregation Level All with Replay Complete for Different Transactions



### 5.5.2 Terminate Subscription

A subscription terminates 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 2.4 Session End on page 6.

## 5.6 Operation <close-session>

Operation <close-session> requests graceful termination of the NETCONF session, see Example 105.

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

*Example 105 Close Session*

## 5.7 Operation <kill-session>

Operation <kill-session> requests forced termination of the selected NETCONF session, see Example 106.

```
<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 106 Kill Session 24*

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

If a NETCONF client receives request <kill-session> for an open session, the NETCONF client performs the following:

- Aborts any operations in progress
- Rolls back the ongoing commit operation
- Discards all uncommitted changes in the transaction
- Releases any locks and resources associated with the session
- Closes the session

Message <ok> is returned if the request is completed without error, otherwise message <rpc-error> is returned indicating the error reason.

Operation `<kill-session>` also triggers transaction abort, as described in Section 9.4 Abort on page 109.

## 5.8 Operation `<copy-config>`

Operation `<copy-config>` 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 parameters are as follows:

- Target – The name of the configuration datastore to use as the destination of operation `<copy-config>`.
- Source – The name of the configuration datastore to use as the source of operation `<copy-config>`, or the `<config>` element containing the complete configuration to copy.

The following 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, an `<rpc-reply>` is sent that includes an `<rpc-error>` element.

### 5.8.1 Error When Target is Same as Source

Operation `<copy-config>` requests arguments of the source datastore and the target datastore to execute the operation. If the same datastore is used for both arguments, the RPC error message in Example 107 is returned.





```
<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 107 Error Message When Target is Same as Source*

## 5.8.2 Error When User Does Not Have Proper Access Right

When a user performs operation `<copy-config>`, the system validates its access right. If the user does not have access, or no rule is configured for this user, or a faulty rule is associated with this user, the RPC error message in Example 108 is returned.

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

*Example 108 Error Message When User Does Not Have Proper Access Right*

## 5.8.3 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 operation `<copy-config>`, the RPC error message in Example 109 or Example 110 is returned.

```
<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 109 Error Message 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 110 Error Message When Source is Set to Candidate Datastore*

## 5.8.4 Error When Empty/Incorrect Namespace is Used

Operation `<copy-config>` requires that the namespace is set to `urn:ietf:params:xml:ns:netconf:base:1.0`. If an empty or incorrect namespace is used, the RPC error message in Example 111 or Example 111 is returned.

```
<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 111 Error Message When Empty Namespace is Used*

```
<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 112 Error Message When Incorrect Namespace "urn:some:namespace" is Used*

### 5.8.5

#### Error When Incorrect Tag is Included as Child Element

Operation `<copy-config>` requires `<source>` and `<target>` as parameters. If an incorrect tag is included as child element in the message, the RPC error message in Example 113 is returned.

```
<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 113 Response when Incorrect Tag is Included as Child Element*

### 5.8.6

#### Error When Invalid Source Datastore is Used

Operation `<copy-config>` requires a valid source configuration datastore, that is, either a running or startup datastore. If an invalid name is used, the RPC error message in Example 114 is returned.

```
<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 114 Error Message When Invalid Source Datastore "rrrrrrrr" is Used*

### 5.8.7

#### Error When `<copy-config>` is Not Supported by Middleware

The `<copy-config>` function closely depends on middleware support. If the middleware does not support the distinct startup operation `<copy-config>`, the RPC error message in Example 115 is returned.

```
<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 115 Error Message When `<copy-config>` is Not Supported by Middleware*

## 5.9

### Operation `<delete-config>`

Operation `<delete-config>` deletes a configuration datastore.



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

### Parameter

This operation has a single parameter:

- **Target** – 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, an `<rpc-reply>` is sent that includes an `<rpc-error>` element.

## 5.9.1 Error When Trying to Delete Running Configuration

According to RFC 6241, the running configuration datastore cannot be deleted. If the running configuration datastore is set as target for operation `<delete-config>`, the RPC error message in Example 116 is returned.

```
<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 116 Error Message When Trying to Delete Running Configuration*

## 5.9.2 Error When Using Inexistent Configuration as Target

Operation `<delete-config>` requires a target datastore to execute the operation. The only supported target is the startup configuration datastore. When inexistent configuration is attempted, the RPC error message in Example 117 is returned.

```
<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 117 Error Message When Inexistent “junkconfig” Configuration is Used as Target*



### 5.9.3 Error When User Does Not Have Proper Access Right

When a user performs operation `<delete-config>`, the system validates its access right. If the user does not have access, or no rule is configured for this user, or a faulty rule is associated with this user, the RPC error message in Example 118 is returned.

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

*Example 118 Error Message When User Does Not Have Proper Access Right*

### 5.9.4 Error When Target is Set to Candidate

Candidate datastore is not supported. If the target argument is set to candidate datastore for operation `<delete-config>`, the RPC error message in Example 119 is returned.

```
<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 119 Error Message When Target is Set to Candidate Datastore*

### 5.9.5 Error When Incorrect Tag is Included as Child Element

Operation `<delete-config>` only requires `<target>` as parameter. If an incorrect tag is included as child element in the message, the RPC error message in Example 120 or Example 121 is returned.

```
<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 120 Error Message When Tag <source> is Included as Child Element*

```
<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 121 Error Message When Inexistent Tag <junkconfig> is Included as Child Element*

### 5.9.6 Error When Adding More than One Child under Tag <target>

Operation `<delete-config>` supports to delete one configuration datastore at a time. If more than one child is added under tag `<target>`, the RPC error message in Example 122 is returned.



```
<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 122 Error Message When Adding More than One Child under Tag <target>*

## 5.9.7

### Error When <delete-config> is Not Supported by Middleware

The <delete-config> function closely depends on middleware support. If the middleware does not support the distinct startup operation <delete-config>, the RPC error message in Example 123 is returned.

```
<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 123 Error Message When <delete-config> is Not Supported by Middleware*





## 6 NETCONF Notifications

The standard-defined notifications `<replayComplete>` and `<notificationComplete>` 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. `<eventTime>` represents the time of the event with the time zone represented in UTC local time in a format compliant to RFC 3339 and ISO 8601, that is, `YYYY-MM-SS'T' hh:mm:ss.+/ -hh' : 'mm`. Fractional seconds are not supported.

Examples of 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)
- 2015-07-02T09:30:00+02:00 during summer, according to Central European Summer Time (CEST)

### 6.1 Notification `<replayComplete>`

Notification `<replayComplete>` in Example 124 is sent after the last notifications requested by a notification replay, as defined in RFC 5277. For details on notification replay, see Section 5.5.1 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 124 Notification `<replayComplete>`*

### 6.2 Notification `<notificationComplete>`

Notification `<notificationComplete>` in Example 125 is sent when `<stopTime>` is reached if it 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 125 Notification `<notificationComplete>`*



## 6.3 Non-Standard Notifications

Non-standard notifications are supported, as described in Section 11.6 New Notifications on page 118.





## 7 NETCONF Validation

After the operation request is received, the NETCONF server automatically performs a validity check. It is done to ensure that the following constitute a configuration complying to the validation rules specified as model properties:

- The configuration entered in the operation
- The running configuration
- The configuration already entered in the transaction

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

### 7.1 Create MO Instance

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

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

```
<?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 126 Error Message When MO Instance Exists*

- 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 127 Error Message for Violation of Cardinality Above Defined Range*

### 7.1.1 Omitting Attributes

The behavior of omitting attributes is 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, if it does not have 0 number of values, and if it is `non-readonly` (`isReadOnly` set to false). (An attribute can have 0 number of values, if it has property `isNillable` in the model or it is a sequence with 0 `minLength`.)

**Note:** The `<mandatory/>` tags in the model are disregarded.

- If an attribute with a 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 7.6.2 Struct Attribute with Default Values on page 92.)

If the default value of an 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.

## 7.2 Delete MO Instance

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 datastore.
- The MO instance is defined as `systemCreated`.
- The number of MO instances after deletion is lower than the minimal cardinality of the MO defined in the model.
- The delete is rejected because of violations of the Model for MOs in a subtree to the deleted 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">MOC: TwoMvStructMv, Cardinality 1 is below =>
[2-3]</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

*Example 128 Error Message for Violation of Cardinality Below Defined Range*



If a delete operation is specified on both the parent MO instance and its child MO instance, the following apply:

- All child MO instances are also deleted during the deletion of the parent MO instance.
- If the child MO instance again tries to delete, the result is an error.

If a delete operation is specified on the parent MO instance and no operation is specified on its child MO instance, the following apply:

- With ebase-0.1.0/ebase-1.1.0 capability use:

After deletion of the parent MO instance, the child MO instance tries to create because of no operation. This results in failure because of a non-existing parent MO instance.

- With ebase-1.2.0 capability use:

Because of the no operation on the child MO, the parent MO operation is inherited. This results in failure during the deletion of the child MO instance, as already child MO instances are deleted during the deletion of the parent MO instance.

## 7.3 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 no 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 defined in the model as `min` and `max` in `lengthRange`.

```
<?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 129 Error Message

- 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 element `validationRules` and attribute `exceptionText` is not empty.

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

```
<?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>validationRulesDateTimeAttr</bad-attribute>
      <bad-element>XThing</bad-element>
    </error-info>
    <error-message xml:lang="en">Data must be time without offset</error-message>
  </rpc-error>
</rpc-reply>
```

#### Example 130 Error Message

- 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 element `validationRules` but attribute `exceptionText` is empty or the string does not comply to the regular expression specified in the deprecated `validValues` element.

**Note:** The format attribute of the rule element, that is, a subelement to element `validationRules`, 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>dateTimeAttr</bad-attribute>
      <bad-element>XThing</bad-element>
    </error-info>
    <error-message xml:lang="en">Invalid value abcdefghijklmnopqrs for attribute dateTimeAttr</error-message>
  </rpc-error>
</rpc-reply>
```

#### Example 131 Error Message

- If the MO attribute is defined as `integer` and the integer value is not in the allowed range that is defined 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 132 Error Message

- If the MO attribute is defined as double and the double value is not in the allowed range that is defined 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 133 Error Message

- If the MO attribute is defined as double and the double value has resolution defined in the model. The attribute must have a range defined in the model as range/min and range/max. The value is invalid if it does not obey the formula  $\text{min} + \text{resolution} \times n \leq \text{max}$ .

```
<?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 134 Error Message

- If the MO attribute is defined as double and the double value has precision higher than 15 digits (this limitation is added to avoid rounding errors for the 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 135 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 136 Error Message

## 7.4 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 datastore.
- 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`.



## 7.5 Change Sequence Attributes

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

- The attribute is defined as sequence and the number of sequence elements is out of the range defined in 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 137 Error Message for Violation of Multiplicity*

- The 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 138 Error Message for Violation of Uniqueness*

## 7.6 Change Struct Attributes

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

- The structure member data types and the number of structure member values are not according to the model definition.
- The structure is defined as exclusive (`isExclusive`) and the structure contains multiple structure members, see Example 139.

Merge of individual struct members on existing singleton structs is supported by the 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 139 Error Message for Violation of Exclusivity*

### 7.6.1 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, the configuration is rejected.
  - If the struct member has a default value, the member is added to the configuration with the default value. If there is no default value, 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` or later is used. See Example 140.
  - 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 141.
  - 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 142.





```
<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 140 Operation merge 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 141 Operation merge 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>
              <aInt32Attr>10</aInt32Attr>
              <aInt64Attr>50</aInt64Attr>
              <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, aInt32Attr MOC: SimpleStructMv </error-message>
  </rpc-error>
</rpc-reply>
]]]]>

```

#### Example 142 Operation merge Failed with Any ebase (Sequence of Structs)

### 7.6.2 Struct Attribute with Default Values

For a struct attribute with default values, the following apply:

- 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, 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. 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.

## 7.7 System-Created Property

Property `isSystemCreated` is replaced by the properties `canCreate` and `canDelete` on `EcimContainment` and `EcimContribution`.

The properties and their respective tags are specified in Table 6.



**Table 6** System-Created Properties and Tags

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

When the `isSystemCreated`, `canCreate`, and `canDelete` properties are true, `propertyisSystemCreated` has the precedence. Otherwise the `canCreate`, and `canDelete` properties have the precedence.

The `canCreate` and `canDelete` properties are used in the following NETCONF operations:

- `create`

Property `canCreate` is used to check whether a MOC is creatable, see Section 7.7.1 Validation for MOC Creation on page 94.

- `delete`

Property `canDelete` is used while deleting a MOC, see Section 7.7.2 Validation for MOC Deletion on page 96.

- `replace (RFC)`

The 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 7.7.3 Validation for MOC replace with RFC Compliance Latest Mode on page 98.

- `<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 4.3 Configuration and State Data on page 16.

Example 143 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 143** Relationship between MOC and Its Parent

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

The error messages are as follows:

- If a MOC has the `<systemCreated/>` tag, the “old” error message is thrown.

Old error message: cannot instantiate “`<mocName>=<instance>`”. The MOC `<mocName>` is system-created.

- If a MOC does not have the `<systemCreated/>` tag, the “new” error message is thrown.

New error message: cannot create “`<mocName>=<instance>`”. The MOC `<mocName>` is not creatable.

The error messages for an invalid `create` operation are shown in Table 7.

**Table 7** Error Messages for Invalid Operation `create`

isSystemCreated	canCreate	canDelete	Error Message
TRUE	TRUE	TRUE	Old
TRUE	FALSE	FALSE	Old
TRUE	FALSE	TRUE	Old



isSystemCreated	canCreate	canDelete	Error Message
FALSE	FALSE	FALSE	New
FALSE	FALSE	TRUE	New

Operation `<edit-config>`, as shown in Example 144, is a request to create MO SimpleMOC. The error messages that are thrown when the previous request is executed are shown in Example 145 and Example 146.

```
<?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 144 Request for Creating a MOC

If SimpleMoc has the `<systemCreated/>` tag, an old error message is thrown as shown in Example 145.

```
<?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>
  </message-id>
</rpc-reply>
]]>]]>
```

#### Example 145 Error Message for Operation create when MOC has <systemCreated/> Tag

If SimpleMoc does not have the `<systemCreated/>` tag, the new error message is thrown as shown in Example 146.

```
<?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 146** Error Message for Operation create when MOC Does Not Have `<systemCreated/>` Tag but `<notCreatable/>` Tag

## 7.7.2 Validation for MOC Deletion

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

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

Error messages are thrown based on the existence of `<systemCreated/>`. The error messages are as follows:

- If the MOC has the `<systemCreated/>` tag, the “old” error message is thrown.

Old error message: cannot delete “`<mocName>=<instance>`”. The MOC `<mocName>` is system-created.

- If the MOC does not have the `<systemCreated/>` tag, the “new” error message is thrown.

New error message: cannot delete “`<mocName>=<instance>`”. The MOC `<mocName>` is not deletable.

The error messages for an invalid `delete` operation are shown in Table 8.

**Table 8** Error Messages for Invalid Operation `delete`

isSystemCreated	canCreate	canDelete	Error Message
TRUE	TRUE	TRUE	Old
TRUE	TRUE	FALSE	Old
TRUE	FALSE	FALSE	Old



isSystemCreated	canCreate	canDelete	Error Message
FALSE	TRUE	FALSE	New
FALSE	FALSE	FALSE	New

Operation `<edit-config>`, as shown in Example 147, is a request to delete MO SimpleMOC. The error messages that are thrown when the previous request is executed are shown in Example 148 and Example 149.

```
<?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 147 Request for Deleting a MOC

If SimpleMoc has the `<systemCreated/>` tag, an old error message is thrown as shown in Example 148.

```
<?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 148 Error Message for Operation delete when MOC Has `<systemCreated/>` Tag

If SimpleMoc does not have the `<systemCreated/>` tag, the new error message is thrown as shown in Example 149.

```
<?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 149 Error Message for Operation delete when MOC Does Not Have <systemCreated/> Tag but <notDeletable/> Tag*

### 7.7.3 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 complies to any of the following:

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

Error messages are thrown based on the existence of `<systemCreated/>`. The error messages are as follows:

- If the MOC has the `<systemCreated/>` tag, the “old” error message is thrown.

Old error message: cannot replace “`<mocName>=<instance>`”. The MOC `<mocName>` is system-created.

- If the MOC does not have the `<systemCreated/>` tag, but contains the `<notDeletable/>` tag, the “deletable” error message is thrown.

Deletable error message: cannot replace “`<mocName>=<instance>`”. The MOC `<mocName>` is not deletable.

- If the MOC does not have the `<systemCreated/>` tag, but contains the `<notCreatable/>` tag, the “creatable” error message is thrown.

Creatable error message: cannot replace “`<mocName>=<instance>`”. The MOC `<mocName>`, is not creatable.

The error messages for invalid `replace` operation are shown in Table 9.





**Table 9 Error Messages for Invalid Operation replace in RFC Compliance Latest Mode**

isSystemCreated	canCreate	canDelete	Error Message
TRUE	TRUE	TRUE	Old
TRUE	TRUE	FALSE	Old
TRUE	FALSE	TRUE	Old
TRUE	FALSE	FALSE	Old
FALSE	TRUE	FALSE	Deletable
FALSE	FALSE	TRUE	Creatable
FALSE	FALSE	FALSE	Deletable

Operation `<edit-config>`, as shown in Example 150, is a request to replace MO SimpleMOC. The error messages that are thrown when the previous request is executed are shown in Example 151, Example 152, and Example 153.

```
<?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 150 Request for Replacing a MOC**

If SimpleMoc has the `<systemCreated/>` tag, an old error message is thrown as shown in Example 151.

```
<?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>
</rpc>]]>]]>
```

**Example 151 Error Message for Operation replace (RFC Compliance Latest Mode) 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 152** Error Message for Operation replace (RFC Compliance Latest Mode) when MOC Does Not Have `<systemCreated/>` Tag but `<notDeletable/>` Tag

If SimpleMoc does not have the `<systemCreated/>` tag, but contains only the `<notCreatable/>` tag, an error message is thrown, as shown in Example 153.

```
<?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 153** Error Message for Operation replace (RFC) when MOC Does Not Have `<systemCreated/>` but Only `<notCreatable/>` Tag

## 7.7.4 Passphrase Property

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

Property `passphrase` and its respective tag are shown are specified in Table 10.

**Table 10** Property `passphrase` and Tag

Property (While Modelling)	Tag (mp.xml File)
<code>isPassphrase=true</code>	<code>&lt;isPassphrase/&gt;</code>

The NETCONF behavior for attributes with property `passphrase` in the NETCONF component file `cfg` is as follows:

- When `<enableEncryptedPassphraseInputOutput/>` is not present in the configuration file, see Table 11.
- When `<enableEncryptedPassphraseInputOutput/>` is present in the configuration file, see Table 12.



**Table 11** NETCONF Behavior for passphrase Attributes in Absence of `<enableEncryptedPassphraseInputOutput/>`

Attribute Type	Setting ( <code>&lt;edit-config&gt;</code> )	Retrieving ( <code>&lt;get&gt;/&lt;get-config&gt;</code> )	Notification Display
KeyAttribute (MOC/struct)	Property is ignored.	Property is ignored and the value is displayed as it is given in the request.  In Example 154, key attribute <code>&lt;passphraseThingId&gt;</code> has property <code>isPassphrase</code> set to <code>true</code> .	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 eight "*" irrespective of the length of the attribute value.	In Example 154, attribute <code>rwattrp</code> has property <code>isPassphrase</code> set to <code>true</code> .	Eight "*" are displayed.  In Example 155, attribute <code>rwattrp</code> has property <code>isPassphrase</code> .
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 property <code>passphrase</code> is ignored and the value is printed as it is.	Not applicable.

**Table 12** NETCONF Behavior for passphrase Attributes in Presence of `<enableEncryptedPassphraseInputOutput/>`

Attribute Type	Setting ( <code>&lt;edit-config&gt;</code> )	Retrieving ( <code>&lt;get&gt;/&lt;get-config&gt;</code> )	Notification Display
KeyAttribute (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 <code>encrypted:{value}</code> , where <code>{value}</code> is in encrypted format (according to an internal algorithm).	
Action parameter/Return value	Property is ignored.	Property is ignored.	Not applicable.

The `<get>` response for attribute containing property `passphrase` is shown in Example 154.



```
<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 154** *<get> Response for Attribute Containing Property passphrase*

The `<notification>` for property attribute `isPassphrase` is shown in Example 155.

```
<?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 155** *<notification> for Property Attribute isPassphrase*

#### 7.7.4.1 Delete Passphrase Attributes

If the delete request contains attribute `passphrase` with value, an RPC error message is returned irrespective of whether the value is present in the database. In Example 156, `rwattrp` has property `isPassphrase` 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 156** *Delete Attribute Passphrase with CDATA Specified in Request*

### 7.7.5 ManagedElement ID Translation

The 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 that if networkManagedElementId is set, the value is retrieved from this attribute, otherwise the value used in the middleware or database is returned.





## 8 NETCONF Visibility Control

The validity of the operations and the returned elements in the reply message depends on the visibility determined by the model and the visibility controller configuration in Life Cycle Management.

The following operations are not allowed to address MOs, MO attributes, or MO action elements if `NOT_VISIBLE` is assigned to their life cycle specifier tag in the configuration file:

- `<get>`
- `<get-config>`
- `<edit-config>`
- `<action>`

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 operation `<edit-config>` or `<get>`. This indicates the directly addressed element that is not visible, while an invisible element like an attribute is skipped in operations `<get>` and `<get-config>`, which are performed on a visible MO containing the invisible attribute. Likewise non-visible MOs in subtrees are skipped when traversing a subtree initiated by operation `<get>` or `<get-config>`.

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







## 9 NETCONF Transaction

Operations in the NETCONF interface are performed through atomic transactions.

### 9.1 Transaction Start

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

### 9.2 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 containing this MO

The MO gets locked in the following cases:

- If an attribute change is requested, the MO containing 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 MOs are locked.
- If an action execution is requested, the MO containing 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 Support Agent or middleware, or other entities providing implementation of a fragment of the model.

## 9.3 Transaction Commit

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

A transaction commit can take place either at the `<close-session>`, at each `<edit-config>`, or at each `<edit-config>` or `<action>` or NEM operation, depending on the ebase behavior of the session. For details, see Section 3.1.2 Capability Identifier ebase on page 9.

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

A failed commit operation is shown in Example 157.

```
<?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 157 Error Message for Operation Commit Failed*

A commit operation fails if a mandatory child MO is not created for the parent MO in the same transaction, see Example 158.

```
<?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=NODE06ST,CMinTop=1,=>
child MOC: RcMin1, Cardinality 0 is below [1-10]</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

### *Example 158 Error Message for Operation Commit Failed Because of Missing Mandatory Child MO*

Failure reason from SA is enclosed in `[]` braces, if Transaction commit has failed as in Example 159.

This information in square braces is displayed only if SA provides it and this gives exact reason why transaction has failed.



```
<?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⇒
[Failed No resources available in SA]</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

*Example 159 Error Message for Transaction Commit Failed Because of no Resource Available in SA*

**Note:** If a parent MOC has a mandatory child MOC with model property `systemCreated`, the mandatory child MO is not validated for existence before commit. In this scenario, commit succeeds.

## 9.4 Abort

A transaction is aborted if the NETCONF session is closed by the events specified in Section 2.4 Session End on page 6, except `close-session` event.

## 9.5 Transaction Time-Out

Transaction time-out value for NETCONF sessions is equal to NETCONF session time-out value.

Transactions in NETCONF session are kept alive by pinging the transaction at regular intervals as long as the NETCONF session is alive.

If NETCONF session is timed out, the current transaction aborts resulting in deletion of transaction content and MO locks.

For example, if the NETCONF session time-out is 5 minutes, the session is closed and transaction is aborted if the server detects no activity in the session during 5 minutes.





## 10 NETCONF Capability <action>

This section describes the Ericsson NETCONF-specific `:action` capability. The description is based on the standard capability template defined in RFC 6241.

### 10.1 Overview

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

### 10.2 Dependencies

None.

### 10.3 Capability Identifier

The action capability is identified by the following capability string:

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

### 10.4 New Operations

This section describes operation `<action>`.

#### 10.4.1 Operation <action>

The NETCONF client sends an action request in message `<rpc>` 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.

##### 10.4.1.1 Operation Request and Parameters

The action request message consists of the following XML elements:

- Element `<rpc>` containing the standard-defined `message-id` and `xmlns` XML attributes.
- Element `<rpc>` contains one `<action>` element with the `xmlns="urn:com:ericsson:ecim:1.0"` namespace definition.
- Element `<action>` contains one `<data>` element.

- Element `<data>` contains the DN of the MO the action is invoked 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 action element name 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 action parameter element name is identical to the action parameter name. Action parameters can be of simple or complex type and are represented according to the notation of the generic Ericsson NETCONF data model. The action parameters can be sent in any sequence order. An action parameter can be omitted if it has a default value defined.

#### 10.4.1.2 Positive Response

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

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

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

- Element `<rpc>` containing the standard-defined `message-id` and `xmlns` XML attributes.
- Element `<rpc>` contains one `<data>` element.
- Element `<data>` contains the DN of the MO the action is invoked 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 action element name is identical to the action name.
- The action element contains one `<returnValue>` element, containing 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 Ericsson NETCONF data model.

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



The start of the action execution can be bound to various conditions depending on the action type, such as the following:

- 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, extra 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.

#### 10.4.1.3 Negative Response

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

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

- `error-type` – Is always `application`.
- `error-tag` – Contains one of the standard-defined values.
- `error-severity` – Is always `error`.
- `error-info` – Identifies the invalid message element.
- `error-message` – Textual description of the error.

#### 10.4.1.4 Examples

An example of an action request is shown in Example 160.

```
<?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>NODE06ST</managedElementId>
        <TestRootMoc>
          <testRootMocId>1</testRootMocId>
          <CallableThing>
            <callableThingId>1</callableThingId>
            <addNumbers>
              <num1>1</num1>
              <num2>2</num2>
            </addNumbers>
          </CallableThing>
        </TestRootMoc>
      </ManagedElement>
    </data>
  </action>
</rpc>]]>]]>
```

### Example 160 Action Request

Examples of multiple return values and a struct return value are shown in Example 161 and Example 162.

```
<?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>NODE06ST</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 161 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>NODE06ST</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 162 Struct Return Value





### 10.4.1.5 Error Messages

Examples of error messages are shown in Example 163 through Example 165.

```
<?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 163 Error Message for 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=NODE06ST,NCTest=1,=>
ActionMoc=1</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

#### *Example 164 Error Message for 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=NODE06ST,=>
SystemFunctions=1,File=1,Logical=1,File=MyFile is not available (no instance).=>
</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

#### *Example 165 Error Message for Non-Existing MO*

## 10.5 Modifications to Existing Operations

None.





# 11 NETCONF Capability <notification>

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

## 11.1 Overview

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

## 11.2 Dependencies

None.

## 11.3 Capability Identifier

The notification capability is identified by the following capability string:

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

## 11.4 New Operations

None.

## 11.5 Modifications to Existing Operations

This section describes the modifications to the existing operations.

### 11.5.1 Operation `<create-subscription>`

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

- The standard-defined parameter `<filter>` of operation `<create-subscription>` contains zero or more Ericsson-specific `<event>` XML elements.
- An `<event>` contains exactly one `<filterType>` element. Valid values of `objectCreated`, `objectDeleted`, and `attributeChanged` means

that a subscription is requested to notifications `<objectCreated>`, `<objectDeleted>`, and `<AVC>`, respectively.

- An `<event>` contains exactly one `<filterValue>` element containing the DN of interest. The name can hold regular expressions in accordance to the POSIX<sup>®</sup> Extended Regular Expressions syntax.
- The namespace of the Ericsson-specific elements `<event>`, `<filterType>`, and `<filterValue>` is `urn:ericsson:com:netconf:notification:1.0`. However, the presence of this namespace is optional and its value is ignored by the Ericsson NETCONF server.

## 11.6 New Notifications

The following Ericsson NETCONF-specific notifications are supported:

- `<CMSynchronizationRecommended>` and `<replayTimeUnsupported>` indicate subscription-related events. These notifications cannot be filtered out and are always sent to the client that has a subscription.
- `<objectDeleted>`, `<objectCreated>`, and `<AVC>` indicate MO changes.
- `<heartbeat>` notifications are periodic notifications sent to the subscriber if the heartbeat capability has been enabled in the session.

### 11.6.1 Notification `<CMSynchronizationRecommended>`

Notification `<CMSynchronizationRecommended>` indicates that one or more MO change-related notifications cannot be sent to the subscribed client. Therefore, synchronization of the configuration and state data is recommended as follows:

- Resending of the lost notifications can be requested by the notification replay request, as described in Section 5.5.1 Create Subscription for Notification Replay on page 71.

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

- System outage
- Notification buffer overflow

This is an internal buffer between the notification service and NETCONF.

- Notification buffer overflow in SA (for CBA users).

When COM SA configured memory limit for notifications handling is reached.

- System clock change



When a clock change event (`MafOamSpiClockChangeEvent`) is received by the notification service, the following happens:

- The replay buffer is cleared.
- `replayLogCreationTime` is set to the time in the event.
- `replayLogAgedTime` is reset.
- Any subsequent events after a clock change event, older than `replayLogCreationTime` are discarded.

The client receiving this notification must retrieve information on the 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 166 Notification <CMSynchronizationRecommended>*

## 11.6.2 Notification <replayTimeUnsupported>

The Ericsson-specific notification `<replayTimeUnsupported>` 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 167 Notification <replayTimeUnsupported>*

## 11.6.3 MO Change Notifications

The following MO change notifications are provided:

- `<objectCreated>`, see Section 11.6.3.1 Notification `<objectCreated>` on page 120
- `<objectDeleted>`, see Section 11.6.3.2 Notification `<objectDeleted>` on page 120
- `<AVC>` (Attribute Value Change), see Section 11.6.3.3 Notification `<AVC>` on page 121

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 a subscription with matching filter type and value.
- The change is applied, that is, the transaction containing 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 if the client initiated the change or not.

The MO change notifications contain the Ericsson NETCONF-specific element `<event>`. It encloses elements `<objectCreated>`, `<objectDeleted>`, and `<AVC>` indicating the type of change.

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

### 11.6.3.1

#### Notification `<objectCreated>`

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

The DN of the created MO is present in XML attribute `dn`.

```
<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=NODE06ST,TestRootMoc=1,BasicThing=2"/>
  </events>
</notification>
]]>]]>
```

#### *Example 168 Notification `<objectCreated>`*

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=NODE06ST,TestRootMoc=1,BasicThing=2"/>
    <attr name="aStringAttr">
      <v>ABC</v>
    </attr>
    <attr name="anInt8Attr">
      <v>42</v>
    </attr>
  </events>
</notification>
]]>]]>
```

#### *Example 169 Notification `<objectCreated>`*

### 11.6.3.2

#### Notification `<objectDeleted>`

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

The DN of the deleted MO is present in XML attribute `dn`.



```
<?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=NODE06ST,TestRootMoc=1,BasicThing=2"/>
  </events>
</notification>
]]>]]>
```

*Example 170 Notification <objectDeleted>*

### 11.6.3.3 Notification <AVC>

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 containing the DN of the changed MO.
- Element <AVC> has one <attr> element.
- Element <attr> has one `name` attribute containing the changed attribute name.
- 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` (multi-value) and has multiple values set, 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` tag, `v` contains a sequence of `elem` elements, each with the struct member name. Each `elem` tag contains at least one `v` tag containing the value of the struct member, in the same way as for attribute members.

When a struct attribute is created, all the struct members are sent in the notification. If the struct members are modified, the notification depends on Middleware interface. If Middleware sends complete struct members along with modified ones, the notification contains all members of the struct. Otherwise, whatever struct members send, only those members are displayed in the notification.

### Examples

Examples of notification <AVC> are shown in Example 171 through Example 177.



```
<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=NODE06ST,TestRootMoc=1,BasicThing=2">
      <attr name="anInt8Attr">
        <v>0</v>
      </attr>
    </AVC>
  </events>
</notification>
]]>]]>
```

#### **Example 171** Notification <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=NODE06ST,TestRootMoc=1">
      <attr name="aBoolAttr">
        <v>false</v>
      </attr>
    </AVC>
  </events>
</notification>
]]>]]>
```

#### **Example 172** Notification <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=NODE06ST,TestRootMoc=1">
      <attr name="aStringMultivalueAttr">
        <v>str1</v>
        <v>str2</v>
        <v>str3</v>
      </attr>
    </AVC>
  </events>
</notification>
]]>]]>
```

#### **Example 173** Notification <AVC> of Sequence (Multi-Value) 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=NODE06ST,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 174** Notification <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=NODE06ST,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 175 Notification <AVC> of Sequence of Structs

If an attribute is defined as `nillable` and all its attribute values are deleted, an <AVC> notification without value element is sent, see Example 176.

```
<?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=NODE06ST">
      <attr name="userLabel"/>
    </AVC>
  </events>
</notification>
]]>]]>
```

#### Example 176 Notification <AVC> of Unset Attribute

If an attribute is defined as `isExclusive`, the struct member with value displays the value set and the other member do not display any value, as it is empty, see Example 177.

```
<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 177 AVC of `isExclusive` struct Attribute

If the client does not specify the `aggregation:1.0` capability, a separate AVC event is sent for each attribute change of an MO, as shown in Example 178.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-27T08:48:19+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <AVC dn="ManagedElement=1,NCTest=1,XThing=2,XXThing=1">
      <attr name="rwattr1">
        <v>aa</v>
      </attr>
    </AVC>
  </events>
</notification>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-27T08:48:19+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <AVC dn="ManagedElement=1,NCTest=1,XThing=2,XXThing=1">
      <attr name="rwattr2">
        <v>15</v>
      </attr>
    </AVC>
  </events>
</notification>
]]>]]>
```

#### Example 178 Separate AVC Notification for Each Attribute Change of an MO

If the client specifies the aggregation:1.0 capability, all attributes change of an MO are sent in a single AVC event, as shown in Example 179.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-27T08:48:19+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <AVC dn="ManagedElement=1,NCTest=1,XThing=2,XXThing=1">
      <attr name="rwattr1">
        <v>aa</v>
      </attr>
      <attr name="rwattr2">
        <v>15</v>
      </attr>
    </AVC>
  </events>
</notification>
]]>]]>
```

#### Example 179 Aggregated AVC Notification for Attribute Change of an MO

### 11.6.4 MO Change Notifications with aggregation:1.0 Capability

The aggregation:1.0 capability introduces the possibility to aggregate multiple CM events (objectCreated, objectDeleted, Attribute Value Change (AVC)) of one transaction into one notification tag.

The Middleware/SA needs to send all the CM events of one transaction as a single CM event notification according to the SPI MafOamSpiCmEvent\_1.h for notifications in a transaction to be aggregated.

#### 11.6.4.1 MO Change Notifications When aggregationLevel = X

When the client specifies the aggregation:1.0 capability and aggregationLevel to an integer value "X" [1,65535], as shown in Example 180 (here the aggregationLevel is 2), "X" events per notification tag are



displayed. If there are 5 events in a transaction and aggregationLevel is 2, then 3 notifications (2 events, 2 events, 1 event) are displayed, as shown in Example 181.

```
<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">
    <aggregationLevel>2</aggregationLevel>
  </create-subscription>
</rpc>
]]>]]>
```

### Example 180 Create-subscription with aggregationLevel 2

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-27T08:12:45+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <objectCreated dn="ManagedElement=1,NCTest=1,XThing=1">
      <attr name="rwattr1">
        <v>RW-One</v>
      </attr>
      <attr name="spacestrattr">
        <v>"abc"</v>
      </attr>
      <attr name="rwattr2">
        <v>1</v>
      </attr>
    </objectCreated>
    <objectDeleted dn="ManagedElement=1,NCTest=1,XThing=2"/>
  </events>
</notification>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-27T08:12:45+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <objectCreated dn="ManagedElement=1,NCTest=1,XThing=3">
      <attr name="rwattr1">
        <v>RW-One</v>
      </attr>
      <attr name="spacestrattr">
        <v>"abc"</v>
      </attr>
      <attr name="rwattr2">
        <v>1</v>
      </attr>
    </objectCreated>
    <objectDeleted dn="ManagedElement=1,NCTest=1,XThing=4"/>
  </events>
</notification>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-27T08:12:45+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <AVC dn="ManagedElement=1,NCTest=1,XThing=5">
      <attr name="rwattr1">
        <v>xx</v>
      </attr>
      <attr name="rwattr2">
        <v>"2"</v>
      </attr>
    </AVC>
  </events>
</notification>
]]>]]>
```

### Example 181 Notification for aggregationLevel=2



#### 11.6.4.2 MO Change Notifications When aggregationLevel=all

When the client specifies the aggregation:1.0 capability and aggregationLevel as “all” in create-subscription, as shown in Example 182, all the events under one transaction are aggregated in one notification, as shown in Example 183.

```
<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">
    <aggregationLevel>all</aggregationLevel>
  </create-subscription>
</rpc>
]]>]]>
```

##### *Example 182 Create-subscription with aggregationLevel all*

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-27T08:17:16+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <objectCreated dn="ManagedElement=1,NCTest=1,XThing=2">
      <attr name="rwattr1">
        <v>RW-One</v>
      </attr>
      <attr name="spacestrattr">
        <v>"abc"</v>
      </attr>
      <attr name="rwattr2">
        <v>1</v>
      </attr>
    </objectCreated>
    <objectDeleted dn="ManagedElement=1,NCTest=1,XThing=3"/>
    <AVC dn="ManagedElement=1,NCTest=1,XThing=2">
      <attr name="rwattr1">
        <v>xx</v>
      </attr>
      <attr name="rwattr2">
        <v>"2"</v>
      </attr>
    </AVC>
  </events>
</notification>
]]>]]>
```

##### *Example 183 All Events of One Transaction are Aggregated in a Single Notification Tag*

#### 11.6.4.3 MO Change Notifications When No aggregationLevel Tag

If only aggregation:1.0 capability is provided and <aggregationLevel> tag is not specified, then aggregationLevel is 1 by default and MO level notifications are displayed as shown in Example 184.



```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-27T08:30:11+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <objectCreated dn="ManagedElement=1,NCTest=1,XThing=2">
      <attr name="rwattr1">
        <v>RW-One</v>
      </attr>
      <attr name="spacestrattr">
        <v>"abc"</v>
      </attr>
      <attr name="rwattr2">
        <v>1</v>
      </attr>
    </objectCreated>
  </events>
</notification>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-27T08:30:11+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <objectDeleted dn="ManagedElement=1,NCTest=1,XThing=3"/>
  </events>
</notification>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-09-27T08:30:11+02:00</eventTime>
  <events dnPrefix="Network=1" xmlns="urn:ericsson:com:netconf:notification:1.0">
    <AVC dn="ManagedElement=1,NCTest=1,XThing=2">
      <attr name="rwattr1">
        <v>xx</v>
      </attr>
      <attr name="rwattr2">
        <v>yy</v>
      </attr>
    </AVC>
  </events>
</notification>
]]>]]>
```

#### Example 184 Notification when Aggregation Level is Not Specified

**Note:** For Heartbeat, ReplayTimeUnsupported, ReplayComplete, NotificationComplete, and CMSynchronizationRecommended events, no aggregations are done.

### 11.6.5 Notification <heartbeat>

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

```
<?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 185 Notification <heartbeat>

## 11.7 XML Schema

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

For complete list of message validity constraints, see Section 11.5.1 Operation `<create-subscription>` on page 117 and Section 11.6 New Notifications on page 118.

### 11.7.1 Notification XML Schema

The XML schema for Ericsson-specific notifications is available in *COMNotification.xsd*.

**Note:** Because of limitations in the XML Schema Definition (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 invalid. 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.

### 11.7.2 Filter Extension XML Schema

The XML schema for the Ericsson-specific `<create-subscription>` filter extension is available in *netconf.xsd*.

*netconf.xsd* imports schema definitions from *COMNotification.xsd*, described in Section 11.7.1 Notification XML Schema on page 128.

According to the schema definition, the possible values of element `<filterType>` are as follows and they are all accepted:

- `ObjectCreated`
- `ObjectDeleted`
- `AttributeChanged`

According to the schema definition, XML attribute `select` can be present in element `<filter>`. However, the attribute is an invalid parameter of operation `<create-subscription>`.

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



## 12 RFC Compliance Latest

The latest RFC-compliant ebase is `urn:com:ericsson:ebase:2.0.0`. See Section 3.1.2 Capability Identifier ebase on page 9.

**Note:** The “RFC compliant latest” ebase is updated every time a new, more RFC compatible ebase is introduced (higher version number means higher compatibility). That means NETCONF ports with “RFC compliant latest” default behavior can change their default behavior when a new ebase is introduced with an upgrade







# 13 NETCONF Statement of Compliance

The Ericsson NETCONF interface is standard compliant according to Table 13.

*Table 13 NETCONF Standard Compliance*

RFC	Compliance
NETCONF protocol Request for Comments (RFCs): <ul style="list-style-type: none"> <li>• RFC 4741 – NETCONF Configuration Protocol <sup>(1)</sup></li> <li>• RFC 6241 – Network Configuration Protocol (NETCONF)</li> </ul>	Partially compliant.  For details, see Table 14 and Section 13.2 NETCONF Compliance to RFC 4741 and RFC 6241 on page 133.
NETCONF over SSH mapping RFCs (these are mandatory to implement): <ul style="list-style-type: none"> <li>• RFC 4742 – Using the NETCONF Configuration Protocol over Secure Shell (SSH) <sup>(2)</sup></li> <li>• RFC 6242 – Using the NETCONF Protocol over Secure Shell (SSH)</li> </ul>	Compliant to RFC 4742.  Partially compliant to RFC 6242.  For details, see Section 13.3 NETCONF Compliance to RFC 4742 and RFC 6242 on page 140.
RFC 5277 – NETCONF Event Notifications	Partially compliant.  For details, see Table 14 and Section 13.4 NETCONF Compliance to RFC 5277 on page 142.
RFC 5539 – NETCONF over Transport Layer Security (TLS)	Partially compliant.  For details, see Section 13.5 NETCONF Compliance to RFC 5539 on page 144.

(1) RFC 4741 is obsoleted by RFC 6241.

(2) RFC 4742 is obsoleted by RFC 6242.

## 13.1 NETCONF Capabilities

A summary of the NETCONF capabilities is shown in Table 14.

Table 14 NETCONF Capabilities

RFC	Capability Identifier	Compliance
4741	urn:ietf:params:netconf:base:1.0	Partially compliant Supported operations: • <get> • <get-config> • <close-session> • <kill-session> • <copy-config> • <delete-config>  Supported with limitation: • <edit-config> <sup>(1)</sup>  Not supported operations: • <lock> <sup>(2)</sup> • <unlock>
6241	urn:ietf:params:netconf:base:1.1	Not compliant
4741, 6241	urn:ietf:params:netconf:capability:writable-running:1.0	Compliant
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 depends on commit behavior <sup>(3)</sup>
4741, 6241	urn:ietf:params:netconf:capability:validate:1.0	Not compliant <sup>(4)</sup>
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	Compliant
6243	urn:ietf:params:netconf:capability:with-defaults:1.0	Not compliant
Non-standard	urn:ericsson:com:netconf:notification:1.0	Ericsson-specific extension
Non-standard	urn:ericsson:com:netconf:action:1.0	Ericsson-specific extension Supported operation: • <action>
Non-standard	urn:com:ericsson:ebase:0.1.0	Ericsson-specific extension
Non-standard	urn:com:ericsson:ebase:1.1.0	Ericsson-specific extension
Non-standard	urn:com:ericsson:ebase:1.2.0	Ericsson-specific extension



Table 14 NETCONF Capabilities

RFC	Capability Identifier	Compliance
Non-standard	urn:com:ericsson:ebase:2.0.0	Ericsson-specific extension
Non-standard	urn:ericsson:com:netconf:heartbeat:1.0	Ericsson-specific extension
Non-standard	urn:com:ericsson:netconf:operation:1.0	Ericsson-specific extension
Non-standard	urn:com:ericsson:netconf:notifications-aggregation:1.0	Ericsson-specific extension

(1) Parameter `<error-option>`, if present, can only be set to `rollback-on-error`. Sending operation `<edit-config>` with a different `<error-option>` results in reply `<rpc-error>` and the ending of the session.

(2) Automatic locking is supported, see Section 9.2 Locking on page 107.

(3) For details, see Section 9.3 Transaction Commit on page 108.

(4) Automatic validation is supported as described in Section 9.2 Locking on page 107.

The following are examples of not supported operations:

- `<commit>`
- `<discard-changes>`
- `<lock>`
- `<unlock>`
- `<validate>`

Not supported operation requests are replied with standard message `operation-not-supported`.

## 13.2 NETCONF Compliance to RFC 4741 and RFC 6241

A summary of the compliance to the following is shown in Table 15:

- RFC 4741 – NETCONF Configuration Protocol  
RFC 4741 is obsoleted by RFC 6241.
- RFC 6241 – Network Configuration Protocol (NETCONF)

Table 15 NETCONF Compliance to RFC 4741 and RFC 6241

RFC Section	Compliance
1. Introduction	Non-normative
1.1. Terminology <sup>(1)</sup>	Non-normative

Table 15 NETCONF Compliance to RFC 4741 and RFC 6241

RFC Section	Compliance
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."
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 in 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



Table 15 NETCONF Compliance to RFC 4741 and RFC 6241

RFC Section	Compliance
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, the Ericsson NETCONF does not use capabilities to announce the set of data models that the ME 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
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"), as described in Section 4 on page 15. 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

Table 15 NETCONF Compliance to RFC 4741 and RFC 6241

RFC Section	Compliance
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
7.2. <edit-config>	<p>Partially compliant</p> <p>Compliant to the operation type requirements of RFC 4741. It includes operations <code>create</code>, <code>delete</code>, <code>replace</code>, and <code>merge</code>.</p> <p>Partially compliant to the operation type requirements of RFC 6241, as option <code>remove</code>, introduced in RFC 6241, is not supported.</p> <p>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")</p> <p>As a limitation, a session is closed if <code>&lt;edit-config&gt;</code> contains unsupported standard defined XML elements or any invalid XML content.</p> <p>For the changes done by <code>&lt;edit-config&gt;</code> the following two commit behaviors are supported:</p> <ul style="list-style-type: none"> <li>• RFC compliant – Commits the changes after each <code>&lt;edit-config&gt;</code>.</li> <li>• Ericsson NETCONF legacy – Commits the changes only when message <code>&lt;close-session&gt;</code> is issued.</li> </ul>



Table 15 NETCONF Compliance to RFC 4741 and RFC 6241

RFC Section	Compliance
7.3. <copy-config>	Partially compliant  Only supports the operation with source as <running> and target as <startup>, as well as with source as <startup> and target as <running>.  As a limitation, the session is closed after the error reply on a non-supported operation that is sent as response
7.4. <delete-config>	Compliant
7.5. <lock>	Not compliant
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

Table 15 NETCONF Compliance to RFC 4741 and RFC 6241

RFC Section	Compliance
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
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 <sup>(1)</sup>	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 operations <copy-config> and <delete-config>.
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





Table 15 NETCONF Compliance to RFC 4741 and RFC 6241

RFC Section	Compliance
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
9. Security Considerations	Compliant  Optional requirement on authorization is supported: "Implementors 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 might 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	Partially 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 <sup>(1)</sup>	Not compliant

Table 15 NETCONF Compliance to RFC 4741 and RFC 6241

RFC Section	Compliance
Appendix D. Capability Template <sup>(2)</sup>	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
D.1.5. Modifications to Existing Operations	Compliant
D.1.6. Interactions with Other Capabilities	Compliant
Appendix E. Configuring Multiple Devices with NETCONF <sup>(3)</sup>	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 <sup>(4)</sup>	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 <sup>(5)</sup>	Non-normative
Appendix G. Changes from RFC 4741 <sup>(6)</sup>	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.

## 13.3 NETCONF Compliance to RFC 4742 and RFC 6242

A summary of the compliance to the following is shown in Table 16:

- 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 16 NETCONF Compliance to RFC 4742 and RFC 6242*

<b>RFC Section</b>	<b>Compliance</b>
1. Introduction	Non-normative
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 <sup>(1)</sup>	Compliant to RFC 6242
4.2. Chunked Framing Mechanism <sup>(1)</sup>	Not compliant to RFC 6242, as the mandatory chunked framing mechanism is not supported
4.3. End-of-Message Framing Mechanism <sup>(1)</sup>	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 <sup>(1)</sup>	Non-normative

*(1) Added in RFC 6242.*

## 13.4 NETCONF Compliance to RFC 5277

A summary of the compliance to RFC 5277 – NETCONF Event Notifications is shown in Table 17.

Table 17 NETCONF Compliance to RFC 5277

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, the Ericsson NETCONF-specific extension to the filter elements is supported instead of the standard filter <subtree> 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



Table 17 NETCONF Compliance to RFC 5277

RFC Section	Compliance
3.2.5.1. Name Retrieval Using <get> Operation	Compliant  As a limitation, Ericsson NETCONF does not support user-specified filters on operation <get>
3.2.5.2. Event Stream Subscription	Compliant
3.2.5.2.1. Filtering Event Stream Contents	Compliant  Only filter type <subtree> 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	Partially compliant  As a limitation, the Ericsson NETCONF-specific extension to the filter elements is 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	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

## 13.5 NETCONF Compliance to RFC 5539

A summary of the compliance to RFC 5539 – NETCONF over Transport Layer Security (TLS) is shown in Table 18.

Table 18 NETCONF Compliance to RFC 5539

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 <sup>(1)</sup> The mandatory-to-implement cipher suite (TLS_RSA_WITH_AES_128_CBC_SHA) is supported
2.2. Connection Closure	Compliant
3. Endpoint Authentication and Identification	Compliant
3.1. Server Identity	Not applicable, as the Ericsson 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.