

COM Management Guide

Common Operation and Maintenance

USER GUIDE

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

1	Introduction	1
2	Fault Management	3
2.1	Basic Concepts	3
2.2	Configuration	6
2.3	Act on Received Notification	8
2.4	Ericsson Alarm MIB	9
2.5	Alarm Information Mapping	10
3	Configuration Management	23
3.1	Basic Concepts	23
3.2	Managed Object Models	23
3.3	Configuration	24
3.4	Configuration Management Using NETCONF	25
3.5	Configuration Management Using CLI	26
3.6	Management of CM Interfaces	26
3.7	Model File Access	30
4	Security Management	31
4.1	Overview	31
4.2	Security Management Functions	33
4.3	Roles and Rules	35
4.4	Configuration	38
5	File Management	41
5.1	Configuration	41
6	Performance Management	43
6.1	Basic Concepts	43
6.2	Measurement Job Administration	44
6.3	Storage of Performance Measurement Results	45
6.4	ECIM PM Model Compliance	47





1 Introduction

This document describes how to use the Fault Management (FM), Configuration Management (CM), Security Management (SM), File Management (FileM), and Performance Management (PM) in the Common Operation and Maintenance (COM) product.





2 Fault Management

The COM FM allows the operator to detect faults and malfunctions on the system. Based on the information carried in the notifications, the operator can locate the source of the event and the relevant documentation. Based on this information, action can be taken to resolve or prevent failures.

2.1 Basic Concepts

Within the COM FM, the following terms are used:

Active alarm	An alarm with an alarm state that has been raised but not cleared.
Active Alarm List	The Active Alarm List (AAL) contains all the active alarms in a system using COM and shows the current operational state of the system. The AAL is stored by the persistent storage provided by the Middleware (MW). The operator can, at any time, retrieve all active alarms through the COM Northbound Interface (NBI) using the Command-Line Interface (CLI), Network Configuration (NETCONF), or Simple Network Management Protocol (SNMP).
Alarm	A persistent indication of the fault that has a life cycle or state. An alarm has at least the state raise (initial detection of the fault) or clear (the detection that the fault no longer exists). An alarm can also change state (update) regarding perceived severity. Alarms are also called “stateful alarms” to emphasize that they have a state.
Alert	A discrete alarm without state, an alarm without an associated clear. Alerts are also called “stateless alarms” to emphasize that they have no state.
Heartbeats	The COM FM sends on regular intervals traps to the Management System. A heartbeat trap contains the last sequence numbers and the last event time used for the alarms and alerts. These messages can be used to detect loss of notifications sent as SNMP traps from the COM FM to the Management System.
Notification	A message that can carry an alarm or alert instance.



Trap An unacknowledged SNMP message that carries a notification or heartbeat.

2.1.1

Logs

The COM FM logs all alarm state changes and all alerts using the Log Service Provider Interface (SPI). Each alarm and alert log record is encoded in XML format.

It is the responsibility of the Support Agent (SA) to map the XML log records on physical files.

The XML log record consists of two elements. The first element indicates the time the record is logged. The second element contains the specific information about the alarm or alert and it is formatted as a string separated by semicolons. The format is explained in Table 1.

For a detailed description of each field, refer to the `FmAlarm` class in *Managed Object Model com_fm*.

Table 1 Alarm and Alert Log Record Format

Tags and Information	Description
<FmLogRecord>	Log record start
<LogTimestamp>	
Time stamp tag	Formatted as YYYY-MM-DDThh:mm:ssZ indicates the time the record is logged ⁽¹⁾
</LogTimestamp>	
<Alarm> or <Alert>	



Tags and Information	Description
Alarm-specific or alert-specific information	<p>Formatted as a string separated by semicolons containing the following tokens:</p> <ul style="list-style-type: none"> 1: stateful 2: eventTime 3: source 4: majorType 5: minorType 6: specificProblem 7: probableCause 8: severity 9: additionalText 10: sequenceNumber 11: eventType 12: originalEventTime 13: originalSeverity 14: originalAdditionalText 15: originalSequenceNumber 16: additionalInfoSize 17: additionalInfo (name;value)
</Alarm> or </Alert>	
</FmLogRecord>	Log record end

(1) Z represents the Coordinated Universal Time (UTC) time zone (equals to +00) or a time zone offset (+hh:mm or –hh:mm).

Examples of log records in an alarm log are shown in Example 1.



```

<FmLogRecord>
  <LogTimestamp>2014-03-27T10:53:55+01:00</LogTimestamp>
  <Alarm>1;2014-03-27T10:53:55+01:00;ManagedElement=1,Equipment=1,⇒
PluginUnit=13;123;429876;HW Fault;418;CRITICAL;Overheated;3;⇒
EQUIPMENTALARM;2014-03-27T10:53:49+01:00;MAJOR;Overheated;1;2;⇒
core1 temp;87;core2temp;93</Alarm>
</FmLogRecord>

<FmLogRecord>
  <LogTimestamp>2014-03-27T10:53:55+01:00</LogTimestamp>
  <Alarm>1;2014-03-27T10:53:55+01:00;ManagedElement=1,Transport=1,⇒
Atm=1;421;792134;Link Overload;613;MAJOR;;5;COMMUNICATIONSALARM;⇒
2014-03-27T10:53:49+01:00;MINOR;Link Overload;2;1;overload;130</Alarm>
</FmLogRecord>

<FmLogRecord>
  <LogTimestamp>2014-03-27T10:53:55+01:00</LogTimestamp>
  <Alarm>1;2014-03-27T10:53:55+01:00;ManagedElement=1,Transport=1,⇒
Atm=1;421;792134;Link overload;613;CLEARED;;5;COMMUNICATIONSALARM;⇒
2014-03-27T10:53:49+01:00;MINOR;Link Overload;2;0</Alarm>
</FmLogRecord>

```

Example 1 Log Records in Alarm Log

If COM has been configured to use a network Managed Element (ME) identity, this is used in the source Distinguished Names (DNs) in log records as shown in Example 2. For more information about network ME identity, see Section 3.3 Configuration on page 24.

```

<FmLogRecord>
  <LogTimestamp>2014-03-27T10:53:55+01:00</LogTimestamp>
  <Alarm>1;2014-03-27T10:53:55+01:00;ManagedElement=Stockholm,⇒
Equipment=1,Plug=13;123;429876;HW Fault;418;CRITICAL;Overheated;⇒
3;EQUIPMENTALARM;2014-03-27T10:53:49+01:00;MAJOR;Overheated;1;2;⇒
core1 temp;87;core2 temp;93</Alarm>
</FmLogRecord>

```

Example 2 Log Record in Alarm Log When Using Managed Element Identity

2.2 Configuration

The COM FM can be configured through the COM NBI using the CLI or NETCONF. The use cases are as follows:

- See Section 2.2.1 Configure SNMP Master Agent on page 7
- See Section 2.2.2 Configure SNMP Master Agent for DTLS on page 7
- Refer to *Create SNMPv1 Target*
- Refer to *Create SNMPv2C Target*



- Refer to *Create SNMPv3 Target*
- Refer to *Enable SNMP Target*
- Refer to *Disable SNMP Target*
- Refer to *Delete SNMP Target*
- Refer to *Create SNMP View*
- Refer to *Check Alarm Status*
- Refer to *Change Alarm Type Severity*
- Refer to *Change Heartbeat Interval* and see Section 2.5.1 Alarm Information in Multiple Interfaces on page 10

For a detailed description of the parameters, refer to *Managed Object Model com_fm* and *Managed Object Model com_snmp*.

2.2.1 Configure SNMP Master Agent

The SNMP Master Agent is configured by changing attributes `agentAddress` in the *Managed Object Model com_snmp*. A CLI example to set the host and port attributes is shown in Example 3.

```
>configure
(config)>ManagedElement=1, SystemFunctions=1, SysM=1, Snmp=1, agentAddress
(config-Snmp=1-agentAddress)>host=127.0.0.1
(config-Snmp=1-agentAddress)>port=161
(config-Snmp=1-agentAddress)>up
(config-Snmp=1)>commit
```

Example 3 Configuring SNMP Master Agent

2.2.2 Configure SNMP Master Agent for DTLS

The SNMP Master Agent is configured for Datagram Transport Layer Security (DTLS) by changing attributes `agentAddressDtls`, `nodeCredential`, and `trustCategory` in the *Managed Object Model com_snmp*. Attributes `nodeCredential` and `trustCategory` give information about certificates. If those attributes are not configured or invalid, SNMP uses the configuration file to get information about the certificates.

A CLI example to configure the Master Agent for DTLS is shown in Example 4.



```
>configure
(config)>ManagedElement=1, SystemFunctions=1, SysM=1, Snmp=1, agentAddressDtls
(config-Snmp=1-agentAddressDtls)>host=0.0.0.0
(config-Snmp=1-agentAddressDtls)>port=10161
(config-Snmp=1-agentAddressDtls)>up
(config-Snmp=1)>nodeCredential=>
ManagedElement=1, SystemFunctions=1, SecM=1, CertM=1, NodeCredential=nc1
(config-Snmp=1)>trustCategory=>
ManagedElement=1, SystemFunctions=1, SecM=1, CertM=1, TrustCategory=tc1
(config-Snmp=1)>commit
```

Example 4 Configuring SNMP Master Agent for DTLS

2.2.3 Create SNMPv3 Target for DTLS

To create a SNMPv3 target for DTLS, attributes `user`, `address`, and `port` must be set.

Attribute `user` must match the `rfc822-name`, Domain Name System (DNS) address, or IP address in the `subjectAltName` portion of the certificate presented by the connecting client. The `CommonName` of the certificate is not used when considering a match, as this is deprecated in the SNMP-TLS-TM Management Information Base (MIB).

A CLI example to create an SNMPv3 target for DTLS is shown in Example 5.

```
>configure
(config)>ManagedElement=1, SystemFunctions=1, SysM=1, Snmp=1, SnmpTargetV3Dtls=1
(config-SnmpTargetV3Dtls=1)>user="user@host.com"
(config-SnmpTargetV3Dtls=1)>address="127.0.0.1"
(config-SnmpTargetV3Dtls=1)>port=10162
(config-SnmpTargetV3Dtls=1)>up
(config-Snmp=1)>commit
```

Example 5 Creating SNMPv3 DTLS Target for DTLS

2.3 Act on Received Notification

Each notification received at the Management System has a corresponding alarm Operating Instruction (OPI). The alarm OPI contains information about the notification and the steps needed to eliminate or prevent failures.

The prerequisites for this procedure are as follows:

- All alarm OPIs must be available.
- The *ERICSSON-ALARM-MIB* document must be available.

To act on a received notification:

1. Find the alarm OPI, whose title is equal to the value of either attribute `eriAlarmActiveSpecificProblem` or `eriAlarmAlertSpecificProblem` of the received notification.
2. Follow the steps in the alarm OPI.



2.4 Ericsson Alarm MIB

The COM FM is compliant with the Ericsson Alarm Management Information Base (MIB) with the following exemptions:

- The alert table, `eriAlarmAlertTable`, is always empty.
- `eriAlarmActiveTableURL` always returns an empty string.
- `eriAlarmAlertTableURL` always returns an empty string.

For more information about the Ericsson Alarm MIB, refer to *ERICSSON-ALARM-MIB*.

2.4.1 AlarmListRebuilt

To comply with the Ericsson Alarm MIB, the COM FM sends an `AlarmListRebuilt` notification after a COM restart and after deactivation of alarm suppression. It is an indication to the operator to perform an alarm resynchronization procedure.

For more information about the notification, refer to *AlarmListRebuilt*.

2.4.2 Alarm Filters

Within the telecom domain, a common problem is that too many alarms are generated. To minimize the number of notifications sent, the COM FM offers two types of alarm filters: alarm toggling filter and transient alarm filter. These filters are by default turned off but can be configured and activated.

2.4.2.1 Alarm Toggling

An alarm is defined as toggling when the same (identical) alarm has been raised and cleared, multiple times, within a defined time.

When the toggling alarm filter is used and an alarm is toggling, alarm attribute `additionalText` is appended with the text "The alarm is toggling". The appended text is kept until the alarm is cleared.

2.4.2.2 Transient Alarm

Short-term and low-priority alarms are considered transient. A high number of transient alarms cause alarm pollution. If the transient alarm filter is used, alarms considered transient are not raised. For more information about alarm filters, refer to *COM Application Developer's Guide*.



2.4.3 Alarm Suppression

The main use cases for alarm suppression are planned maintenance tasks where the alarm client risks are flooded with non-important alarms. One example is when a technician is present on-site handling the equipment. All alarms are still available in the AAL, but no alarm notifications are sent.

However, alarms with critical severity, for example, fire alarms, are sent regardless. Active critical alarms that are cleared or assigned a lower severity also generate a notification to indicate that the critical problem is resolved.

For more information on how to turn alarm suppression on/off, refer to *COM API and SPI Programmer's Guide*.

2.5 Alarm Information Mapping

2.5.1 Alarm Information in Multiple Interfaces

Alarm information is available in the SNMP, CLI, and NETCONF interfaces and in log files, as described in this section.

The heartbeat interval of the COM SNMP alarm interface is configurable as follows:

- SNMP object `eriAlarmHbInterval` of the *ERICSSON-ALARM-MIB*
- Attribute `heartbeatInterval` of the `Fm` Managed Object (MO) in the COM CLI or NETCONF

A summary of the number of active alarms is available as follows:

- `eriAlarmActiveAlarms`, `eriAlarmSumCritical`, `eriAlarmSumMajor`, `eriAlarmSumMinor`, and `eriAlarmSumWarning` SNMP objects of the *ERICSSON-ALARM-MIB*

Note: The number of alarms of indeterminate severity is available in the SNMP as `eriAlarmSumIndeterminate` only.

- Attributes `totalActive`, `sumCritical`, `sumMajor`, `sumMinor`, and `sumWarning` of MO `Fm` in the COM CLI and NETCONF

Note: While reading the `Fm` subtree in a particular transaction, the statistics attributes in `Fm` (that is `totalActive`, `sumMajor`, `sumMinor`, and so on) cannot always be in sync with the actual `FmAlarm` instances.

Information about the last issued alarm is available as follows:

- SNMP objects `eriAlarmActiveLastSequenceNo` and `eriAlarmActiveLastChanged` of the *ERICSSON-ALARM-MIB*



- Attributes `lastChanged` and `lastSequenceNo` of MO Fm in the COM CLI and NETCONF

Alarms instances are available in multiple interfaces, as summarized in Table 2.

Table 2 Representations of Alarms

Alarm and Alert Log	FmAlarm MO	SNMP Object	SNMP Notification
eventTime	lastEventTime	eriAlarmActiveEventTime, eriAlarmActiveOriginalEventTime	eriAlarmActiveEventTime
source	source	eriAlarmActiveManagedObject	eriAlarmActiveManagedObject
majorType	majorType	eriAlarmActiveMajorType	eriAlarmActiveMajorType
minorType	minorType	eriAlarmActiveMinorType	eriAlarmActiveMinorType
specificProblem	specificProblem	eriAlarmActiveSpecificProblem	eriAlarmActiveSpecificProblem
probableCause	probableCause	eriAlarmActiveProbableCause	eriAlarmActiveProbableCause
severity	activeSeverity	eriAlarmActiveSeverity, eriAlarmActiveOriginalSeverity	The notification type indicates severity information.
additionalText	additionalText	eriAlarmActiveAdditionalText, eriAlarmActiveOrigAdditionalText	eriAlarmNObjAdditionalText, eriAlarmNObjMoreAdditionalText
sequenceNumber	sequenceNumber	eriAlarmActiveLastSequenceNo	eriAlarmActiveLastSequenceNo
eventType	eventType	eriAlarmActiveEventType	eriAlarmActiveEventType
Not applicable	additionalInfo	Not applicable	Not applicable
Not applicable	fmAlarmId	Not applicable	Not applicable
Not applicable	Not applicable	eriAlarmActiveResourceId	eriAlarmNObjResourceId

(1) The notification types are `eriAlarmCritical`, `eriAlarmMajor`, `eriAlarmMinor`, `eriAlarmWarning`, `eriAlarmIndeterminate`, and `eriAlarmCleared`.

2.5.2 SNMP Objects

Properties of the supported SNMP objects are summarized in Table 3. The deviations are indicated.

Table 3 Alarm Objects Defined in Ericsson Alarm MIB

Object Name and Properties	Description
ericssonAlarmMIB OID: .1.3.6.1.4.1.193.183.4	
eriAlarmObjects OID: .1.3.6.1.4.1.193.183.4.1	
eriAlarmSummary Type: Gauge32 OID: .1.3.6.1.4.1.193.183.4.1.1 Max Access: read-only	



Table 3 Alarm Objects Defined in Ericsson Alarm MIB

Object Name and Properties	Description
eriAlarmSumIndeterminate Type: Gauge32 OID: .1.3.6.1.4.1.193.183.4.1.1.1 Max Access: read-only	This object shows the number of currently active alarms with perceived severity <i>indeterminate</i> . Only stateful alarms are included.
eriAlarmSumCritical Type: Gauge32 OID: .1.3.6.1.4.1.193.183.4.1.1.2 Max Access: read-only	This object shows the number of currently active alarms with perceived severity <i>critical</i> . Only stateful alarms are included.
eriAlarmSumMajor Type: Gauge32 OID: .1.3.6.1.4.1.193.183.4.1.1.3 Max Access: read-only	This object shows the number of currently active alarms with perceived severity <i>major</i> . Only stateful alarms are included.
eriAlarmSumMinor Type: Gauge32 OID: .1.3.6.1.4.1.193.183.4.1.1.4 Max Access: read-only	This object shows the number of currently active alarms with perceived severity <i>minor</i> . Only stateful alarms are included.
eriAlarmSumWarning Type: Gauge32 OID: .1.3.6.1.4.1.193.183.4.1.1.5 Max Access: read-only	This object shows the number of currently active alarms with perceived severity <i>warning</i> . Only stateful alarms are included.
eriAlarmActiveAlarms OID: .1.3.6.1.4.1.193.183.4.1.3	
eriAlarmActiveNumber Type: Unsigned32 OID: .1.3.6.1.4.1.193.183.4.1.3.1 Max Access: read-only	This object shows the total number of currently active alarms, that is, the total number of entries in the AAL.
eriAlarmActiveLastChanged Type: DateAndTime OID: .1.3.6.1.4.1.193.183.4.1.3.2 Max Access: read-only	A time stamp when the active AAL was last changed. The value can be used by an operator to initiate an alarm resynchronization procedure. All fields of <i>DateAndTime</i> must be filled out, including the hours and minutes from Coordinated Universal Time (UTC). As such, the value must be 11 octets long. Reference: <i>DateAndTime</i> is defined in RFC 2579.
eriAlarmActiveLastSequenceNumber Type: EriAlarmSequenceNumber OID: .1.3.6.1.4.1.193.183.4.1.3.3 Max Access: read-only	The last used sequence number for an alarm state change notification. A Management System can poll this variable to detect lost alarm change notifications.



Table 3 Alarm Objects Defined in Ericsson Alarm MIB

Object Name and Properties	Description
eriAlarmActiveTableURL Type: SnmpAdminString OID: .1.3.6.1.4.1.193.183.4.1.3.4	<p>A URL, in accordance with RFC 4248, pointing to a location where the contents of the alarm table (which is the same as AAL) can be retrieved as a file. The actual file format is out of the scope of this MIB and is found in the system documentation. A string of length zero means that the agent has no URL to provide.</p> <p>Deviation: This object always contains an empty string.</p>
eriAlarmActiveMajorType Type: EriAlarmType OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.2 Max Access: read-only	<p>In combination with <code>eriAlarmActiveMinorType</code>, this provides a unique identification of the fault type. Different MO types and instances can share alarm types, but if the same MO reports the same alarm type, it is to be considered as the same alarm state. The alarm type is a simplification of the different X.733 and 3GPP® alarm Integration Reference Point (IRP) alarm correlation mechanisms based on <code>EventType</code>, <code>ProbableCause</code>, <code>SpecificProblem</code>, and <code>NotificationId</code>. In systems where <code>eriAlarmActiveMajorType</code> is not needed for identification purposes, it is not used and must be zero.</p>
eriAlarmActiveMinorType Type: EriAlarmType OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.3 Max Access: read-only	<p>In combination with <code>eriAlarmActiveMajorType</code>, this provides a unique identification of the fault type, not including the MO. Different MO types and instances can share alarm types, but if the same MO reports the same alarm type, it is to be considered as the same alarm state. The alarm type is a simplification of the different X.733 and 3GPP alarm IRP alarm correlation mechanisms based on <code>EventType</code>, <code>ProbableCause</code>, <code>SpecificProblem</code>, and <code>NotificationId</code>.</p>
eriAlarmActiveSpecificProblem Type: EriAlarmSpecificProblem OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.4 Max Access: read-only	<p>A clear-text unique identification of the alarm type. The mapping between <code>eriAlarmActiveSpecificProblem</code> and <code>[eriAlarmActiveMajorType, eriAlarmActiveMinorType]</code> is one-to-one.</p>
eriAlarmActiveManagedObject Type: EriMO OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.5 Max Access: read-only	<p>The 3GPP naming convention must be used as format for the MO parameter. The granularity must be good enough to guarantee unique alarm states and relevant resource identification to the operator.</p> <p>The DN must be “relative” to the nodes “own” root.</p> <p>Reference: 3GPP TS 32.106-8 V3.2, Name convention for Managed Objects.</p>

Table 3 Alarm Objects Defined in Ericsson Alarm MIB

Object Name and Properties	Description
eriAlarmActiveEventType Type: IANAItuEventType OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.6 Max Access: read-only	The event type as defined in X.733/X.736. Reference: ITU Recommendation X.733, "Information Technology – Open Systems Interconnection – System Management: Alarm Reporting Function", 1992.
eriAlarmActiveEventTime Type: DateAndTime OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.7 Max Access: read-only	A time stamp of the alarm state change event. This variable represents the last change of the alarm state, such as changed severity or additional text. If the alarm has not changed state, this variable represents the alarm raise time and is the same as <code>originalEventTime</code> . All fields of <code>DateAndTime</code> must be filled out, including the hours and minutes from UTC. As such, the value must be 11 octets long. References: <code>DateAndTime</code> is defined in RFC 2579. ITU Recommendation X.733, "Information Technology – Open Systems Interconnection – System Management: Alarm Reporting Function", 1992.
eriAlarmActiveOriginalEventTime Type: DateAndTime OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.8 Max Access: read-only	The time stamp of the original alarm raises a notification. All fields of <code>DateAndTime</code> must be filled out, including the hours and minutes from UTC. As such, the value must be 11 octets long. Reference: <code>DateAndTime</code> is defined in RFC 2579.
eriAlarmActiveProbableCause Type: EriProbableCause OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.9 Max Access: read-only	The probable cause for the alarm originally defined by X.733 and subsequent standards. Refer also to the <i>ERICSSON-ALARM-PC-MIB</i> . Because of the history of problems in maintaining a standardized probable cause, the probable cause is not unique. A best effort mapping of the alarm to existing probable causes is used. Reference: ITU Recommendation X.733, "Information Technology – Open Systems Interconnection – System Management: Alarm Reporting Function", 1992.
eriAlarmActiveSeverity Type: ItuPerceivedSeverity OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.10 Max Access: read-only	The severity of the alarm as defined by X.733. This cannot be the original severity, as the alarm has changed severity. Reference: ITU Recommendation X.733, "Information Technology – Open Systems Interconnection – System Management: Alarm Reporting Function", 1992.



Table 3 Alarm Objects Defined in Ericsson Alarm MIB

Object Name and Properties	Description
eriAlarmActiveOriginalSeverity Type: ItuPerceivedSeverity OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.11 Max Access: read-only	The original severity as reported by the alarm raise notification.
eriAlarmActiveAdditionalText Type: EriLargeAdditionalText OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.12 Max Access: read-only	A user-friendly text describing the alarm. The text is both static, depending on the alarm type (probable cause), and dynamic depending on the MO instance and other conditions. This string is longer than the corresponding varbind in the notification to manage large strings. Reference: ITU Recommendation X.733, "Information Technology – Open Systems Interconnection – System Management: Alarm Reporting Function", 1992.
eriAlarmActiveOrigAdditionalText Type: EriLargeAdditionalText OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.13 Max Access: read-only	A user-friendly text describing the alarm. The text is both static, depending on the alarm type (probable cause), and dynamic depending on the MO and other conditions. This is the original text as reported by the first alarm notification, after which it is not altered. Reference: ITU Recommendation X.733, "Information Technology – Open Systems Interconnection – System Management: Alarm Reporting Function", 1992.
eriAlarmActiveResourceId OID: .1.3.6.1.4.1.193.183.4.1.3.5.1.14 Max Access: read-only	If the alarm refers to an object that is instrumented by the SNMP, this variable indicates the corresponding OID for the MO. Deviation: Not supported. The value is null OID (0.0).
eriAlarmAlerts OID: .1.3.6.1.4.1.193.183.4.1.4	
eriAlarmAlertNumber Type: Unsigned32 OID: .1.3.6.1.4.1.193.183.4.1.4.1 Max Access: read-only	The number of rows in the alert table.
eriAlarmAlertLastChanged Type: DateAndTime OID: .1.3.6.1.4.1.193.183.4.1.4.2 Max Access: read-only	A time stamp when the alert table was last changed. Can be used by an operator to initiate an update of alerts procedure. All fields of DateAndTime must be filled out, including the hours and minutes from UTC. As such, the value must be 11 octets long. Reference: DateAndTime is defined in RFC 2579.



Table 3 Alarm Objects Defined in Ericsson Alarm MIB

Object Name and Properties	Description
eriAlarmAlertLastSequenceNo Type: EriAlarmSequenceNumber OID: .1.3.6.1.4.1.193.183.4.1.4.3 Max Access: read-only	The last sequence number used in alert notifications. This can be used to detect a lost notification.
eriAlarmAlertTableURL OID: .1.3.6.1.4.1.193.183.4.1.4.4 Max Access: read-only	A URL, in accordance with RFC 4248, pointing to a location where the contents of alert table can be retrieved as a file. The file format is out of the scope of this MIB and system-dependant. Deviation: This object always contains an empty string.
eriAlarmAlertTable OID: .1.3.6.1.4.1.193.183.4.1.4.5	Deviation: The alert table is always empty.
eriAlarmHeartBeat OID: .1.3.6.1.4.1.193.183.4.1.5 Max Access: read-only	
eriAlarmHbInterval Type: Unsigned32 OID: .1.3.6.1.4.1.193.183.4.1.5.1 Max Access: read-write	Notification <code>eriAlarmHeartBeatNotif</code> is sent every <code>eriAlarmHbInterval</code> . Operators can subscribe to the notification using the SNMP framework MIBs by using the <code>snmpNotifyName</code> "heartbeat" (SNMP NOTIFICATION MIB, <code>snmpNotifyTable</code>).
eriAlarmNObjAdditionalText Type: EriAdditionalText OID: .1.3.6.1.4.1.193.183.4.1.2.1 Max Access: accessible-for-notify	A scalar variable, only used in notifications. It is size-ranged with a smaller size than the corresponding type used in the active AAL. This is to ensure that all alarm information fits into one User Datagram Protocol (UDP) packet. The additional text can be further appended by using the dedicated append notification.
eriAlarmNObjMoreAdditionalText Type: TruthValue OID: .1.3.6.1.4.1.193.183.4.1.2.2 Max Access: accessible-for-notify	A scalar variable, only used in notifications. It informs the Management System that the additional text is appended by subsequent append additional text notifications. This object exists because of the limitations of the Protocol Data Unit (PDU) size inherent in the SNMP over UDP, which limits the size of the PDU+headers to under 1500 octets. Otherwise, the notification becomes fragmented.
eriAlarmNObjResourceId Type: TruthValue OID: .1.3.6.1.4.1.193.183.4.1.2.3 Max Access: accessible-for-notify	A scalar variable only, used in notifications. It tells the Management System that there is an SNMP-based resource ID (OID) that identifies the alarming resource. This object exists because of the limitations of the PDU size inherent in the SNMP over UDP, which limits the size of the PDU+headers to under 1500 octets. Otherwise, the notification becomes fragmented. Deviation: Not supported. Always equals 0.



2.5.3 SNMP Notifications

Properties of the supported SNMP notifications of the *ERICSSON-ALARM-MIB* are summarized in Table 4. The deviations are indicated.

Table 4 Notification Objects Defined in Ericsson Alarm MIB

Object Name and Properties	Description
eriAlarmIndeterminate Type: { eriAlarmActiveManagedObject, eriAlarmActiveMajorType, eriAlarmActiveMinorType, eriAlarmActiveSpecificProblem, eriAlarmActiveLastSequenceNo, eriAlarmActiveEventType, eriAlarmActiveEventTime, eriAlarmActiveProbableCause, eriAlarmNObjAdditionalText, eriAlarmNObjMoreAdditionalText, eriAlarmNObjResourceId } OID: .1.3.6.1.4.1.193.183.4.2.0.1	<p>This notification is sent when a resource detects a new alarm state with severity <i>indeterminate</i>. The notification is also used to change the severity or additional text of an alarm, or both. The combination of <i>ManagedObject</i> and <i>MajorType/MinorType</i> is always unique and can be used by Management Systems to correlate alarm, alarm change, and alarm clear. A corresponding row is created in the AAL. The sequence number increases for every notification and can be used to detect lost notifications.</p> <p>A Management System is to be prepared for appending text to additional text, indicated by the <i>eriAlarmNObjMoreAdditionalText</i> varbind, and sent with <i>eriAlarmAppendInfo</i>.</p> <p>Do not confuse this with a change of additional text.</p> <p>A Management System is also to be prepared to receive a resource ID (OID) identifying the alarming resource if the system sending the notification can provide that information. In that case, <i>eriAlarmNObjResourceId</i> is set to <i>true</i> and the resource ID is sent in an <i>eriAlarmAppendInfo</i> notification.</p>



Table 4 Notification Objects Defined in Ericsson Alarm MIB

Object Name and Properties	Description
eriAlarmWarning Type: { eriAlarmActiveManagedObject, eriAlarmActiveMajorType, eriAlarmActiveMinorType, eriAlarmActiveSpecificProblem, eriAlarmActiveLastSequenceNo, eriAlarmActiveEventType, eriAlarmActiveEventTime, eriAlarmActiveProbableCause, eriAlarmNObjAdditionalText, eriAlarmNObjMoreAdditionalText, eriAlarmNObjResourceId } OID: .1.3.6.1.4.1.193.183.4.2.0.2	<p>This notification is sent when a resource detects a new alarm state with severity <code>warning</code>. The notification is also used to change the severity or additional text of an alarm, or both. The combination of <code>ManagedObject</code> and <code>MajorType/MinorType</code> is always unique and can be used by Management Systems to correlate alarm, alarm change, and alarm clear. A corresponding row is created in the alarm table (<code>eriAlarmActiveAlarmTable</code>). Alarm table is the same as AAL. The sequence number increases for every notification and can be used to detect lost notifications.</p> <p>A Management System is to be prepared for appending text to additional text, indicated by the <code>eriAlarmNObjMoreAdditionalText</code> varbind, and sent with <code>eriAlarmAppendInfo</code>.</p> <p>Do not confuse this with a change of additional text.</p> <p>A Management System is also to be prepared to receive a resource ID (OID) identifying the alarming resource if the system sending the notification can provide that information. In that case, <code>eriAlarmNObjResourceId</code> is set to <code>true</code> and the resource ID is sent in an <code>eriAlarmAppendInfo</code> notification.</p>



Table 4 Notification Objects Defined in Ericsson Alarm MIB

Object Name and Properties	Description
eriAlarmMinor Type: { eriAlarmActiveManagedObject, eriAlarmActiveMajorType, eriAlarmActiveMinorType, eriAlarmActiveSpecificProblem, eriAlarmActiveLastSequenceNo, eriAlarmActiveEventType, eriAlarmActiveEventTime, eriAlarmActiveProbableCause, eriAlarmNObjAdditionalText, eriAlarmNObjMoreAdditionalText, eriAlarmNObjResourceId } OID: .1.3.6.1.4.1.193.183.4.2.0.3	<p>This notification is sent when a resource detects a new alarm state with severity <code>minor</code>. The notification is also used to change the severity or additional text of an alarm, or both. The combination of <code>ManagedObject</code> and <code>MajorType/MinorType</code> is always unique and can be used by Management Systems to correlate alarm, alarm change, and alarm clear. A corresponding row is created in the alarm table (<code>eriAlarmActiveAlarmTable</code>). The sequence number increases for every notification and can be used to detect lost notifications.</p> <p>A Management System is to be prepared for appending text to additional text, indicated by the <code>eriAlarmNObjMoreAdditionalText</code> varbind, and sent with <code>eriAlarmAppendInfo</code>.</p> <p>Do not confuse this with a change of additional text.</p> <p>A Management System is also to be prepared to receive a resource ID (OID) identifying the alarming resource if the system sending the notification can provide that information. In that case, <code>eriAlarmNObjResourceId</code> is set to <code>true</code> and the resource ID is sent in an <code>eriAlarmAppendInfo</code> notification.</p>
eriAlarmMajor Type: { eriAlarmActiveManagedObject, eriAlarmActiveMajorType, eriAlarmActiveMinorType, eriAlarmActiveSpecificProblem, eriAlarmActiveLastSequenceNo, eriAlarmActiveEventType, eriAlarmActiveEventTime, eriAlarmActiveProbableCause, eriAlarmNObjAdditionalText, eriAlarmNObjMoreAdditionalText, eriAlarmNObjResourceId } OID: .1.3.6.1.4.1.193.183.4.2.0.4	<p>This notification is sent when a resource detects a new alarm state with severity <code>major</code>. The notification is also used to change the severity or additional text of an alarm, or both. The combination of <code>ManagedObject</code> and <code>MajorType/MinorType</code> is always unique and can be used by Management Systems to correlate alarm, alarm change, and alarm clear. A corresponding row is created in the alarm table (<code>eriAlarmActiveAlarmTable</code>). The sequence number increases for every notification and can be used to detect lost notifications.</p> <p>A Management System is to be prepared for appending text to additional text, indicated by the <code>eriAlarmNObjMoreAdditionalText</code> varbind, and sent with <code>eriAlarmAppendInfo</code>.</p> <p>Do not confuse this with a change of additional text.</p> <p>A Management System is also to be prepared to receive a resource ID (OID) identifying the alarming resource if the system sending the notification can provide that information. In that case, <code>eriAlarmNObjResourceId</code> is set to <code>true</code> and the resource ID is sent in an <code>eriAlarmAppendInfo</code> notification.</p>



Table 4 Notification Objects Defined in Ericsson Alarm MIB

Object Name and Properties	Description
eriAlarmCritical Type: { eriAlarmActiveManagedObject, eriAlarmActiveMajorType, eriAlarmActiveMinorType, eriAlarmActiveSpecificProblem, eriAlarmActiveLastSequenceNo, eriAlarmActiveEventType, eriAlarmActiveEventTime, eriAlarmActiveProbableCause, eriAlarmNObjAdditionalText, eriAlarmNObjMoreAdditionalText, eriAlarmNObjResourceId } OID: .1.3.6.1.4.1.193.183.4.2.0.5	<p>This notification is sent when a resource detects a new alarm state with severity <code>critical</code>. The notification is also used to change the severity or additional text of an alarm, or both. The combination of <code>ManagedObject</code> and <code>MajorType/MinorType</code> is always unique and can be used by Management Systems to correlate alarm, alarm change, and alarm clear. A corresponding row is created in the alarm table (<code>eriAlarmActiveAlarmTable</code>). The sequence number increases for every notification and can be used to detect lost notifications.</p> <p>A Management System is to be prepared for appending text to additional text, indicated by the <code>eriAlarmNObjMoreAdditionalText</code> varbind, and sent with <code>eriAlarmAppendInfo</code>.</p> <p>Do not confuse this with a change of additional text.</p> <p>A Management System is also to be prepared to receive a resource ID (OID) identifying the alarming resource if the system sending the notification can provide that information. In that case, <code>eriAlarmNObjResourceId</code> is set to <code>true</code> and the resource ID is sent in an <code>eriAlarmAppendInfo</code> notification.</p>
eriAlarmCleared Type: { eriAlarmActiveManagedObject, eriAlarmActiveMajorType, eriAlarmActiveMinorType, eriAlarmActiveSpecificProblem, eriAlarmActiveLastSequenceNo, eriAlarmActiveEventType, eriAlarmActiveEventTime, eriAlarmActiveProbableCause, eriAlarmNObjAdditionalText, eriAlarmNObjMoreAdditionalText, eriAlarmNObjResourceId } OID: .1.3.6.1.4.1.193.183.4.2.0.7	<p>This notification is sent when a resource detects a cleared alarm state. The combination of <code>ManagedObject</code> and <code>MajorType/MinorType</code> is always unique and must be used by Management Systems to correlate alarm and alarm clear. The corresponding row in the alarm table is deleted (<code>eriAlarmActiveAlarmTable</code>). The sequence number increases for every notification and can be used to detect lost notifications.</p> <p>A Management System is to be prepared for appending text to additional text, indicated by the <code>eriAlarmNObjMoreAdditionalText</code> varbind, and sent with <code>eriAlarmAppendInfo</code>.</p> <p>Do not confuse this with a change of additional text.</p> <p>A Management System is also to be prepared to receive a resource ID (OID) identifying the alarming resource if the system sending the notification can provide that information. In that case, <code>eriAlarmNObjResourceId</code> is set to <code>true</code> and the resource ID is sent in an <code>eriAlarmAppendInfo</code> notification.</p>



Table 4 Notification Objects Defined in Ericsson Alarm MIB

Object Name and Properties	Description
eriAlarmAppendInfo Type: { eriAlarmActiveManagedObject, eriAlarmActiveMajorType, eriAlarmActiveMinorType, eriAlarmNObjAdditionalText, eriAlarmActiveResourceId } OID: .1.3.6.1.4.1.193.183.4.2.0.8	<p>This notification is sent to append further information to an existing alarm. It can be additional text or a resource ID (OID), identifying the alarming resource using an OID.</p> <p>If additional text is sent, do not confuse this with an actual change of additional text, which is reported using the eriAlarm severity notification.</p> <p>A zero-length string value for eriAlarmNObjAdditionalText means that no additional text is sent in this notification.</p> <p>A null OID (0.0) value for eriAlarmActiveResourceId means that no resource ID is sent in this notification.</p>
eriAlarmHeartBeatNotif Type: { eriAlarmActiveLastSequenceNo, eriAlarmAlertLastSequenceNo, eriAlarmActiveLastChanged, eriAlarmAlertLastChanged } OID: .1.3.6.1.4.1.193.183.4.2.0.20	<p>A heartbeat notification with interval according to eriAlarmHbInterval. It contains the last sequence numbers used for alarms and alarm events. These varbinds can be used to detect lost notifications. The notification eriAlarmHeartBeatNotif is sent every eriAlarmHbInterval. Operators can subscribe to the notification using the SNMP framework MIBs by using the snmpNotifyName "heartbeat" (SNMP NOTIFICATION MIB, snmpNotifyTable).</p>
eriAlarmAlarmListRebuilt Type: { eriAlarmActiveTableURL } OID: .1.3.6.1.4.1.193.183.4.2.0.30	<p>This notification is sent when the AAL has reached a stable situation after a restart or after a system-internal audit process. It is an indication to the operator to perform an alarm resynchronization procedure.</p> <p>Deviation: Not supported. eriAlarmActiveTableURL always returns an empty string.</p>

In the real notifications, OIDs for all varbinds belonging to eriAlarmActiveAlarmEntry or eriAlarmActiveAlertEntry are appended with the corresponding unique index of alarms or alerts in eriAlarmActiveAlarmTable or eriAlarmActiveAlertTable respectively. For example, OID of varbind eriAlarmActiveManagedObject in an alarm notification corresponding to index 4 in eriAlarmActiveAlarmTable ends with a 4, that is .1.3.6.1.4.1.193.183.4.1.3.5.1.5.4.

The same is applicable for other varbinds belonging to eriAlarmActiveAlarmTable or eriAlarmActiveAlertTable.





3 Configuration Management

The COM CM allows the operator to view and change the configuration data in the system.

3.1 Basic Concepts

Within the COM CM, the following term is used:

MIM File The structure of the configuration data is specified in terms of Management Information Model (MIM) files. The MIM XML files correspond to the Managed Object Models (MOMs) that are expressed using the UML®. The MIM XML files comply with the Ericsson-internal Document Type Definition (DTD) named `mp.dtd`. COM uses the MIM files to validate CM requests.

The COM Runtime provides a tool to add, delete, and upgrade model files. For a detailed description, refer to *COM Application Developer's Guide*.

3.1.1 Logs

COM separates logging of important system events from debugging information. Audit information is also generated. To separate the log information from the debugging information, the NBI Agents use the Log SPI and the Trace SPI offered by COM.

For more information and exact location of the log files, refer to the documentation of the MW SA for each specific MW.

3.2 Managed Object Models

The COM CM function delivers the MOMs specified in Table 5. These MOMs constitute a true subset of the Ericsson Common Information Model (ECIM). Other application-specific MOMs are also accessible from the COM NBI, but these are out of scope of this document.

Table 5 Managed Object Models

Managed Object Model	Description
Refer to <i>Managed Object Model com_top</i> .	Defines the top-level structure of the entire MO class hierarchy.



Managed Object Model	Description
Refer to <i>Managed Object Model com_sysm</i>	Defines the model that includes MO classes for the system-level functions.
Refer to <i>Managed Object Model com_fm</i> .	Defines the model that includes MO classes for the alarm model.
Refer to <i>Managed Object Model com_snmp</i> .	Defines the model that includes MO classes for the SNMP interface.
Refer to <i>Managed Object Model com_secm_local_authorization</i> .	Defines the model that includes MO classes for the local authorization model.
Refer to <i>Managed Object Model com_filem</i> .	Defines the model that includes MO classes for file management.

3.3 Configuration

COM System Management can be configured through the COM NBI using the CLI or NETCONF. The use cases are as follows:

- See Section 3.3.1 Change 'ManagedElementId' on page 24
- See Section 3.6.1 Configure CLI over SSH on page 26
- See Section 3.6.2 Configure CLI over TLS on page 27
- See Section 3.6.3 Configure NETCONF over SSH on page 28
- See Section 3.6.4 Configure NETCONF over TLS on page 28
- Refer to *Check General System Attributes*
- Refer to *List Supported XML Model Files*
- Refer to *Push XML Model File to Remote File Location*

For a detailed description of the parameters, refer to *Managed Object Model com_sysm*.

3.3.1 Change 'ManagedElementId'

The identity of the root MO, `ManagedElement`, is set at factory to some default value, normally 1. In a live network, the identity must be set to a unique value provided by the operator. The most important reasons for this are to be able to identify the Network Element (NE) in alarms, PM data, logs, and so on, and to be able to connect the NE MIB into the overall network MIB.



The naming attribute (identity) of an MO cannot be changed after creation, therefore COM provides an additional attribute on `ManagedElement` called `networkManagedElementId`. This attribute, if set, is used as the `ManagedElement` identity, replacing attribute `managedElementId`. If set to an empty string, the `managedElementId` is used.

The `networkManagedElementId` is intended to be used during commissioning of an NE, but the attribute can be changed at any time. If it is changed in runtime, all alarms in the AAL have the source MO updated so, but other data or files that have been generated in the system are not updated.

An example of factory setting is shown in Example 6.

```
* managedElementId=1
* networkManagedElementId=""
* Valid MO DN: ManagedElement=1, SystemFunctions=1, Fm=1
```

Example 6 Factory Setting

An example of live network settings is shown in Example 7.

```
* managedElementId=1
* networkManagedElementId=Stockholm
* Valid MO DN: ManagedElement=Stockholm, SystemFunctions=1, Fm=1
```

Example 7 Live Network Settings

To change `ManagedElementId`:

1. Log on to the COM CLI, refer to *Ericsson Command-Line Interface*.
2. Enter Config mode:

```
>configure
```

3. Change `networkManagedElementId` to the desired value:

```
(config)>ManagedElement=1,networkManagedElementId=<new value>
```

4. Commit changes and return to Exec mode:

```
(config)>commit
```

3.4 Configuration Management Using NETCONF

NETCONF is an XML-based protocol that defines operations for accessing and updating a configuration data store.

For detailed information on how to use the NETCONF interface, refer to *Ericsson NETCONF Interface*.



3.5 Configuration Management Using CLI

The COM CLI is an interface to the Operations, Administration, and Maintenance (OAM) model. The model has a tree-like structure with a `ManagedElement` as the root, and the subparts and their attributes as child nodes.

When logging on to the CLI from the Secure Shell (SSH), a session is started with the current location placed at a top node, named the ghost node, which is just above the root `ManagedElement` node.

Nodes are identified by their type name and instance ID. The instance ID is usually (by convention) a number, but can be any string. For example, a field or node storing the serial number of a board can be called `serialNumber=1`.

A field is equivalent to an attribute. A node is equivalent to an MO.

For detailed information on how to use the CLI, refer to *Ericsson Command-Line Interface*.

3.6 Management of CM Interfaces

This section describes how to configure CLI and NETCONF services.

3.6.1 Configure CLI over SSH

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Use this section only if the SSH management feature is supported by the product.

CLI over SSH can be enabled or disabled by changing attribute `administrativeState` (LOCKED/UNLOCKED) of the `CliSsh` class in the *Managed Object Model* `com_sysm`.

The behavior of CLI over SSH based on the state of attribute `administrativeState` is defined as follows:

- If the `administrativeState=LOCKED`:
 - All existing SSH-based CLI connections are closed.
 - No new SSH-based CLI connections are accepted.
- If `administrativeState=UNLOCKED`:
 - Users are allowed to make new connections.
- `administrativeState` defaults to UNLOCKED.



3.6.2 Configure CLI over TLS

Note: INSTRUCTION FOR DOCUMENT REUSE: Use this section only if the Transport Layer Security Daemon (TLS Daemon) management feature is supported by the product. For configuration of TLS Daemon, refer to Section *COM TLS Daemon Manager Configuration File* in *COM Application Developer's Guide*.

COM CLI over TLS is configured for certificates by changing attributes `nodeCredential` and `trustCategory` of the `CliTls` class in the *Managed Object Model com_sysm*.

The configuration of `CliTls` is invalid in the following case:

- When attribute `trustCategory` or `nodeCredential` is empty and the other refers to respective MO of the `CertM` MOM.

Example 8 shows how to configure MO `CliTls` using the CLI.

```
>configure
(config)>ManagedElement=1, SystemFunctions=1, SysM=1, CliTls=1
(config-CliTls=1)>nodeCredential="ManagedElement=1, SystemFunctions=1, SecM=1, =>
CertM=1, NodeCredential=1"
(config-CliTls=1)>trustCategory="ManagedElement=1, SystemFunctions=1, SecM=1, =>
CertM=1, TrustCategory=1"
(config-CliTls=1)>commit
```

Example 8 Configuring `CliTls` MO Using CLI

Example 9 shows how to configure MO `CliTls` using NETCONF.

```
<?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 xmlns="urn:com:ericsson:ecim:ComTop">
          <systemFunctionsId>1</systemFunctionsId>
          <SysM xmlns="urn:com:ericsson:ecim:ComSysM">
            <sysMId>1</sysMId>
            <CliTls xmlns="urn:com:ericsson:ecim:ComSysM" xc:operation="merge">
              <cliTlsId>1</cliTlsId>
              <nodeCredential>ManagedElement=1, SystemFunctions=1, SecM=1, CertM=1, =>
NodeCredential=1</nodeCredential>
              <trustCategory>ManagedElement=1, SystemFunctions=1, SecM=1, CertM=1, =>
TrustCategory=1</trustCategory>
            </CliTls>
          </SysM>
        </SystemFunctions>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>
]]>]]>
```

Example 9 Configuring `CliTls` MO Using NETCONF



3.6.2.1 Administrative State

Attribute `administrativeState` of the `CliTls` class controls the CLI connections over Transport Layer Security (TLS). The behavior of CLI over TLS based on the state of attribute `administrativeState` is defined as follows:

- If `administrativeState=LOCKED`:
 - All existing TLS-based CLI connections are closed.
 - No new TLS-based CLI connections are accepted.
- If `administrativeState=UNLOCKED`:
 - Users are allowed to make new connections.
- `administrativeState` defaults to `LOCKED`.

3.6.3 Configure NETCONF over SSH

Note: **INSTRUCTION FOR DOCUMENT REUSE:** Use this section only if the SSH management feature is supported by the product.

NETCONF over SSH can be enabled or disabled by changing attribute `administrativeState` (`LOCKED/UNLOCKED`) of the `NetconfSsh` class in the *Managed Object Model* `com_sysm`.

The behavior of NETCONF over SSH based on the state of attribute `administrativeState` is defined as follows:

- If `administrativeState=LOCKED`:
 - All existing SSH-based NETCONF connections are closed.
 - No new SSH-based NETCONF connections are accepted.
- If `administrativeState=UNLOCKED`:
 - Users are allowed to make new connections.
- `administrativeState` defaults to `UNLOCKED`.

3.6.4 Configure NETCONF over TLS

COM NETCONF over TLS is configured for certificates by changing attributes `nodeCredential` and `trustCategory` of the `NetconfTls` class in the *Managed Object Model* `com_sysm`. If these attributes are not configured, the NETCONF TLS component configuration file is used for certificate information.



The configuration of NETCONF TLS is invalid in the following case:

- When attribute `trustCategory` or `nodeCredential` is empty and the other refers to respective MO of the `CertM` MOM.

Note: Credential User Application Programming Interface (API) is required to use certificates from the `CertM` MOM for NETCONF over TLS. If not found, the NETCONF TLS component configuration file is used for certificates.

Example 10 shows how to configure MO `NetconfTls` using the CLI.

```
>configure
(config)>ManagedElement=1, SystemFunctions=1, SysM=1, NetconfTls=1
(config-NetconfTls=1)>nodeCredential="ManagedElement=1, SystemFunctions=1, =>
SecM=1, CertM=1, NodeCredential=1"
(config-NetconfTls=1)>trustCategory="ManagedElement=1, SystemFunctions=1, =>
SecM=1, CertM=1, TrustCategory=1"
(config-NetconfTls=1)>commit
```

Example 10 Configuring NetconfTls MO Using CLI

Example 11 shows how to configure MO `NetconfTls` using NETCONF.

```
<?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 xmlns="urn:com:ericsson:ecim:ComTop">
          <systemFunctionsId>1</systemFunctionsId>
          <SysM xmlns="urn:com:ericsson:ecim:ComSysM">
            <sysMid>1</sysMid>
            <NetconfTls xmlns="urn:com:ericsson:ecim:ComSysM" xc:operation="merge">
              <netconfTlsId>1</netconfTlsId>
              <nodeCredential>"ManagedElement=1, SystemFunctions=1, SecM=1, CertM=1, =>
NodeCredential=1"</nodeCredential>
              <trustCategory>"ManagedElement=1, SystemFunctions=1, SecM=1, CertM=1, =>
TrustCategory=1"</trustCategory>
            </NetconfTls>
          </SysM>
        </SystemFunctions>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>
]]>]]>
```

Example 11 Configuring NetconfTls MO Using NETCONF

3.6.4.1 Administrative State

Attribute `administrativeState` of the `NetconfTls` class controls the NETCONF connections over TLS. The behavior of NETCONF over TLS based on the state of attribute `administrativeState` is defined as follows:

- If `administrativeState=LOCKED`:
 - All existing TLS-based NETCONF connections are closed.



- No new TLS-based NETCONF connections are accepted.
- If `administrativeState=UNLOCKED`:
 - Users are allowed to make new connections.
- `administrativeState` defaults to `LOCKED`.

3.7 Model File Access

COM provides access to the MIM files loaded by COM. Data related to the MIM files is available through the NBI, according to the `Schema` class of the *Managed Object Model* `com_sysm`. In runtime, one `Schema` instance exists for each configured MIM file.

The MIM files themselves can be downloaded with the export action of the `Schema` instances.

4 Security Management

This section describes the SM capabilities of COM and how to use them.

4.1 Overview

The COM SM consists of authorization support of the MOM for users accessing through the NBI. The MOM authorization is based on a combination of roles and rules. User authentication is performed by the SSH.

For an overview of COM SM, see Figure 1.

The flow used for authentication of a user is shown in Table 6.

Table 6 Authentication and Authorization Flow

Authentication	1. SSH authenticates the O&M user by using a Pluggable Authentication Module (PAM).	
	2. SSH passes the user identity to the COM NBI Agent.	
Authorization	3. The COM NBI agent indicates the start of a new user session to COM SM.	
	4. COM retrieves the roles of the O&M user from the central Authorization Information Store (AIS).	4A. The local store is checked first to see if it is an emergency user (refer to Section <i>Create Emergency Group</i> in <i>COM SW Installation Base</i>). COM uses the local database under all conditions. COM considers the Linux <code>/etc/passwd</code> , <code>/etc/group</code> , and <code>/etc/shadow</code> files as the local AIS.
		4B. If the Access Management (AM) SPI v1 is provided, COM turns to the MW SA to get the roles.
		4C. Otherwise COM SM directs the search to the AM component. COM AM uses the Access Control Service (ACS) User Access Information (UAI) API to retrieve the roles.
	5. COM retrieves the rules for the roles of the O&M user from the local AIS.	
	6. For each O&M operation, the COM NBI agent asks COM SM if access can be granted for the O&M user on the provided operation.	

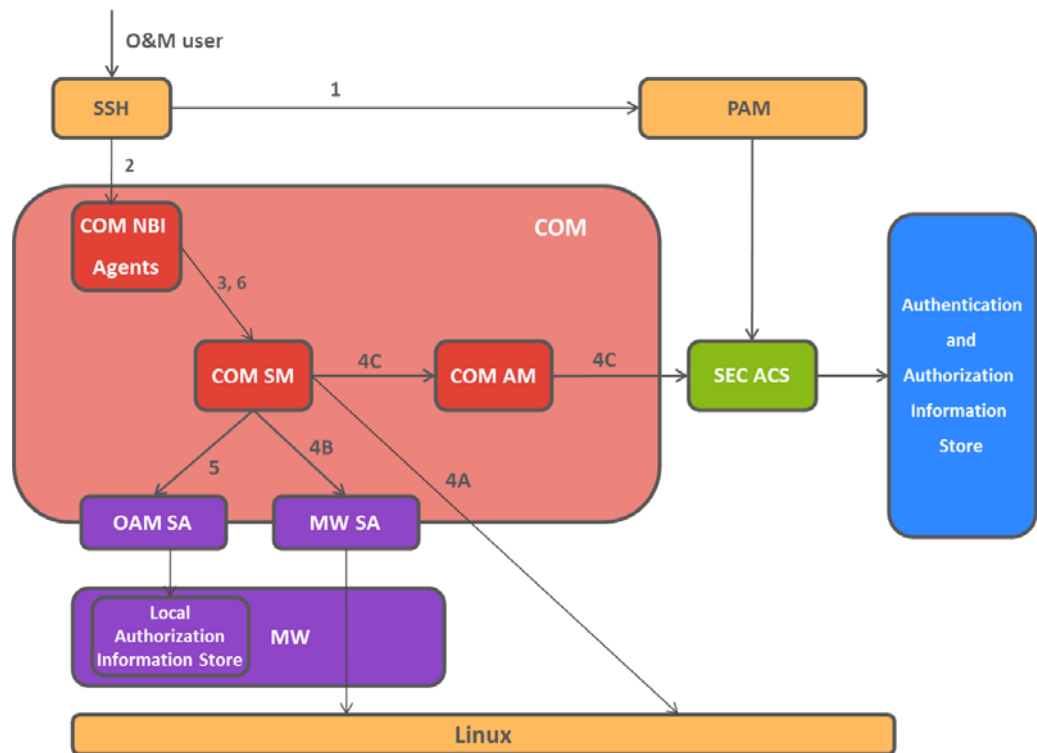


Figure 1 SM within COM

4.1.1 Authorization Support of MOM for Users Accessing NBI

The roles for a user are stored in authentication data storage.

4.1.2 OS Level Authentication and Authorization

The classical OS-level authentication and authorization is handled by the OS layer. Examples are file-level permission, binary execute permissions, and storage media permissions.

4.1.3 SM Behavior

This section describes the SM behavior from a user perspective.

4.1.3.1 Administrative States

The administrative states are as follows:

- When the SM service is in locked state, no authorization is performed. Hence all MO operations, MO actions, and CLI commands are allowed.



Note: The terms “SM service in locked state” and “SM service in unlocked state” means the local state in the SM component based on the `administrativeState` for the ECIM authorization MO.

- When the SM service is in unlocked state, authorization is performed. The change of the `administrativeState` affects existing and new sessions.

4.1.3.2

General SM Behavior

The general SM behavior is characterized by the following:

- At the start of every user NBI Agent session, SM data for that session (user roles, rules) is read from the authorization data storage.

The SM data is updated during the session if an O&M user with the appropriate privileges changes any or a combination of the following in the *Managed Object Model* `com_secm_local_authorization`:

- Attribute `administrativeState`
- Class `CustomRole`
- Class `CustomRule`
- Class `Role`
- Class `Rule`
- When the SM MIB is not initialized, the SM service allows any user to execute any NBI Agent operation.
- Authorization exceptions: If any exceptions occur during authorization rule evaluation, such as rule syntax error, the operation is denied. The exception is logged in the system error log by the SM component.
- Access management failure in retrieving roles: If the SM service is unlocked, all MO operations, MO actions, and CLI commands are denied, as role data is not available.

4.2

Security Management Functions

The conceptual view of the user authentication and authorization data is shown in Figure 2.

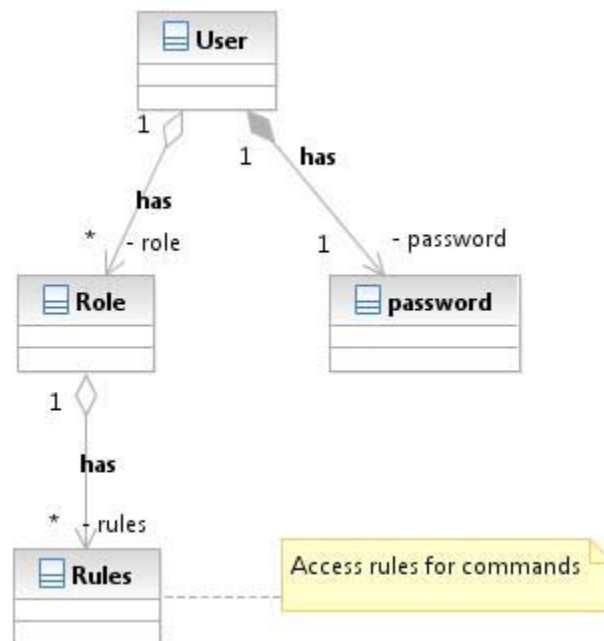


Figure 2 User Authentication and Authorization Data

4.2.1 Fault Management

The SM sends no notifications.

4.2.2 Audit Logging

Audit logging (that is, who logs on and off from the system) is not performed by the SM. The MW or OS layer authenticates the NBI user.

4.2.3 Audit Trail Logging

Audit trail logging (that is, who has done what and when) is not performed by the SM. Instead, it is performed by the NBI Agents, using the Log SPI.

4.2.4 Capacity and Resource Consumption

The resource consumption depends on the size of the role and rule data in the authorization model.

4.2.5 Response Time

The response time depends on the authenticate data storage failover time.



4.3 Roles and Rules

COM delivers role instances for the following:

- `SystemAdministrator`
- `SystemSecurityAdministrator`
- `SystemReadOnly`
- `SystemTroubleshooter`
- `Self`

COM also delivers the rule instances connected to the following roles:

- `SystemAdministrator`
- `SystemSecurityAdministrator`
- `SystemReadOnly`

These rule instances only cover the MOM fragments delivered by COM, thus other components and applications must contribute with their own rules. COM does not deliver any rules for the `Self` and `SystemTroubleshooter` roles.

For more details on these roles, see Section 4.3.1 Default Roles and Rules on page 35.

COM supports implicit navigation through parent fragments. There is no need to specify read access to parent fragments explicitly. Thus the syntax '`MOC1=abc,MOC2=pqr,...,MOCN=xyz$`' for `ruleData` is deprecated. This is illustrated as follows:

```
ReadWriteExecute access to SecM:
  Rule name: SecurityManagement_1
  Permission: ReadWriteExecute
  Rule data: ManagedElement, SystemFunctions, SecM, *
```

Access is granted as follows:

- Read access to the `ManagedElement` MO, but not its attributes. Enables navigation in the CLI.
- Read access to the `SystemFunctions` MO, but not its attributes. Enables navigation in the CLI.
- RWX access to the `SecM` MO and all MOs below, including all attributes and actions.

4.3.1 Default Roles and Rules

4.3.1.1 System Administrator

A system administrator is responsible for the administration of all non-security-related attributes and capabilities of an ME, for example, ME features, capabilities, configuration parameters, and monitoring of the ME.



The following rules are delivered by COM to provide the system administrator with proper access to model fragments supported by COM:

- Full access to the `ManagedElement` MO and its attributes.
- Full access to the `SystemFunctions` MO and its attributes.
- Full access to the `Transport` MO and its attributes.
- Full access to the `Fm` MO and all MOs below, including all attributes and actions.
- Full access to the `SysM` MO and all MOs below, including all attributes and actions.

4.3.1.2 System Security Administrator

A system security administrator is responsible for the administration of the attributes and capabilities of an ME related to security of the ME, regardless of which applications that execute on the ME, for example, ME administrative, user accounts, and authorizations.

The following rules are delivered by COM to provide the system security administrator with proper access to model fragments supported by COM:

- Read access to the `ManagedElement` MO, but not its attributes. Enables navigation in the CLI.
- Read access to the `SystemFunctions` MO, but not its attributes. Enables navigation in the CLI.
- Read access to the `Fm` MO and all MOs below, including all attributes.
- Full access to the `SecM` MO and all MOs below, including all attributes and actions.

4.3.1.3 System Read Only

A system read only is responsible for the monitoring of system configuration parameters.

The following rules are delivered by COM to provide the system security administrator with proper access to model fragments supported by COM:

- Read access to the `ManagedElement` MO and its attributes.
- Read access to the `SystemFunctions` MO and its attributes.
- Read access to the `Transport` MO and its attributes.
- Read access to the `Fm` MO and all MOs below, including all attributes.



- Read access to the `sysM` MO and all MOs below, including all attributes.

4.3.1.4

Self

COM provides one predefined role `Self` for every user because of which default authorization is enabled for applications. This role is assigned to every user by default.

COM does not deliver any rules for the role `Self`. Applications that are interested in “Self authorization” can configure application-specific rules under the `Self` role.

Rules defined under the `Role` MO instance `Self` are automatically assigned to the session of a user.

COM supports parameterized and non-parameterized rules under the `Self` role. See Example 12 for a parameterized rule.

For local users, COM applies both parameterized and non-parameterized rules available under the `Self` role. For remote users, only non-parameterized rules under the `Self` role are applied.

Before applying rules under the `Self` role to a user, COM uses SEC Access Control Service (ACS) User Access Information (UAI) to find whether that user is “local” or not. So, to use the “Self authorization” feature, SEC ACS UAI must be available for COM.

```
RuleName : testParameterizedRule
RuleData : ManagedElement=1, SystemFunctions=1, SecM=1, UserManagement=1, UserAccountM=#u, SshPublicKey
Permission : RW
```

Example 12 Parameterized Rule

Here, `#u` is the parameter. For now, `username` is the only parameter considered. See Table 7.

Table 7 Parameters in Parameterized Rule

Parameter	Parameter Description	Parameter Notation
<code>username⁽¹⁾</code>	The username of the session. The username is asserted by the service of authentication.	<code>#u</code>

(1) This parameter can give access to MO resources related to user accounts in authentication methods. To achieve that, the authentication method declares itself as “local” to authorize the policy decision point for parameter substitution.

The simplified regular expression of `RuleDataType` only works for key attributes.



There must be a case-sensitive exact string match between the session parameters and the naming attribute of the specified MO instance.

Note: No roles other than `Self` must contain parameterized rules.

COM restricts NBI users to configure the `CustomRole` MO instance with the role name `Self`.

4.4 Configuration

The COM SM can be configured through the COM NBI using the CLI or NETCONF. The use cases are as follows:

- Refer to *Unlock Local Authorization Method* and see Section 4.4.1 Change Administrative State Using NETCONF on page 38
- Refer to *Lock Local Authorization Method*
- Refer to *Create Custom Role*
- Refer to *Change Custom Role*
- Refer to *Delete Custom Role*
- Refer to *Create Custom Rule*
- Refer to *Change Custom Rule*
- Refer to *Delete Custom Rule*
- Refer to *View Roles and Rules*

For a detailed description of the parameters, refer to *Managed Object Model com_secm_local_authorization*.

The prerequisites for these procedures are as follows:

- The user must be assigned a system authentication role having sufficient privileges to access the SM branch.
- `ManagedElement`, `SystemFunctions`, `SecM`, `UserManagement`, and `LocalAuthorizationMethod` must exist with instance ID = 1. Attribute `networkManagedElementId` of `ManagedElement` must be unset or set to 1.

4.4.1 Change Administrative State Using NETCONF

To change `administrativeState` using NETCONF, in this case setting the SM authorization to state `UNLOCKED`:

1. Use a NETCONF client to send the following to the NE where COM executes:



```
<?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>
          <SecM xmlns="urn:com:ericsson:ecim:ComSecM">
            <secMId>1</secMId>
            <UserManagement>
              <userManagementId>1</userManagementId>
              <LocalAuthorizationMethod =>
                <localId>1</localId>
                <administrativeState>UNLOCKED</administrativeState>
              </LocalAuthorizationMethod>
            </UserManagement>
          </SecM>
        </SystemFunctions>
      </ManagedElement>
    </config>
  </edit-config>
</rpc>
```

Example 13 Changing Administrative State Using NETCONF





5 File Management

FileM is a service implementing parts of the ECIM fragment `FileM` and it is characterized by the following:

- Provides a way for file producers and file consumers on the ME to organize and maintain (housekeep) files and folders in the file system.
- Provides and facilitates services to expose files and folders to NBI services, for example, the SSH File Transfer Protocol (SFTP), CLI, and NETCONF.

Note: COM supports DN-restricted characters “+”, “*”, and “#” in the `FileGroup` value, which are displayed as `\2B`, `\2A`, and `\23` in the NBI, where 2B, 2A, and 23 are the hexadecimal ASCII code of these characters, except these all the other DN-restricted characters (<, >, ; and so on) in the `FileGroup` value are displayed as “??” in the NBI. All the DN-restricted characters (+, <, >, #, and so on) in the Filenames are displayed as “??” in the NBI.

For MO name convention and a full list of DN-restricted characters, refer to [3GPP TS 32.300 V9.0.0](#).

All access to files and folders through the NBI is authorized by rules defined in the ECIM `SecM` fragment, see Section 4 on page 31.

5.1 Configuration

The COM FileM can be configured through the COM NBI using the CLI or NETCONF. The use cases are as follows:

- Refer to *Fetch File in Logical File System*
- Refer to *Delete File in Logical File System*
- Refer to *List File Groups and File Information in Logical File System*
- Refer to *Configure Preventive Maintenance Policy Deleting Files in Logical File System*
- Refer to *Configure Preventive Maintenance Policy Reporting Alarms for Logical File System*

For a detailed description of the parameters, refer to *Managed Object Model `com_filem`*.





6 Performance Management

3GPP TS 32.401 V10.0.0 defines PM as follows:

“Any evaluation of a system’s behavior requires performance data collected and recorded by its NEs according to a schedule established by the NE manager. This aspect of the management environment is termed Performance Management. The purpose of any Performance Management activity is to collect data, which can be used to verify the physical and logical configuration of the network and to locate potential problems as early as possible.”

TS 32.401 also defines the different phases in the process of performance measurement administration. COM is responsible of two of them: measurement job administration (section 4.2.1 in the TS) and local storage of the results at the NE (section 4.2.3 in the TS). Generation of the performance measurement results is the responsibility of the MW system.

6.1 Basic Concepts

3GPP TS 32.401 describes all the PM concepts. The most relevant concepts from a COM user point of view are as follows:

Job granularity period

Time between the initiation of two successive gatherings of measurement data triggered by this job.

Job reporting period

Time between the generations of two report files including measurement data generated by this job.

Measurement job

Process executed in the NE to accumulate measurement result data and assemble it for collection or inspection, or both.

ROP report file

The Result Output Period (ROP) report file is an XML file compliant with the 3GPP TS 32.432 V10.0.0. The file contains the results of one or more measurement collections triggered by one or more measurement jobs along one or more Granularity Periods (GPs).



6.2 Measurement Job Administration

The ECIM PM fragment provides support to perform all the tasks related with the administration of measurement jobs. The `Pm` class is present under `SystemFunctions` and this is present under `ManagedElement`.

6.2.1 PmJob

Measurement jobs are represented as instances of `PmJob` defined in the ECIM PM fragment. Administration of jobs (creation, deletion, and modification of job attributes) is made by accessing the ECIM through the NBI (through the CLI or NETCONF) and instantiating and deleting `PmJob` MOs, or changing the values of the attributes in these MOs.

COM requires ECIM PM Version 1.2 or higher. For information about the meaning of the attributes and possible values, refer to *ECIM PM Documentation*.

Deviations in COM from ECIM PM Version 1.2 are as follows:

- COM PM assumes that the Job Reporting Period is equal to the Job GP.
- COM PM does not support job priority.
- COM PM does not limit the number of measurements in a report.

6.2.2 Configure PmGroup

Several instances of `PmGroup` can be found under the `Pm` class. They describe the measurements that are available on the NE.

According to the ECIM PM fragment documentation “*The PmGroup is a grouping of the measurements into logical grouping*”. Each `PmGroup` has one or more `MeasurementType` instances.

To add a measurement to a certain job:

1. Create a `MeasurementReader` MO under the `PmJob` MO.
2. Set the members of struct `measurementSpecification` so they contain a reference to the `PmGroup` and `MeasurementType` to be added to the job.

6.2.3 PmMeasurementCapabilities

The `PmMeasurementCapabilities` class of the ECIM PM fragment contains attributes describing the measurement capabilities of the current NE. For details of the meaning of each attribute, refer to *ECIM PM Documentation*.



6.3 Storage of Performance Measurement Results

Once the measurement jobs are set up, the MW system is responsible of collecting the measurement results for every job. Once the results are ready, at the end of the GP of every measurement job, the COM PM functionality writes results into Result Output Period (ROP) report files.

6.3.1 Location of ROP Files

The location where the ROP files are stored can be retrieved from attribute `fileLocation` in the `PmMeasurementCapabilities` class.

If `FileM` is used:

- `fileLocation` is the relative path `/PerformanceManagementReportFiles`.
- `fileGroup` is the DN of a `FileGroup` MO.

If `FileM` is not used:

- `fileLocation` is the absolute path, for example: `/var/filem/internal_root/PerformanceManagementReportFiles`.
- `fileGroup` is empty.

The COM PM functionality performs housekeeping of PM report files after each GP ends. The number of reports published does not exceed the value of attribute `maxNoOfPmFiles` in `PmMeasurementCapabilities`.

Note: It is not allowed to set up `FileM` housekeeping policies of PM report files or the PM report files directories. Such a configuration leads to undefined behavior in the system. Attribute `fileGroup` is deprecated from ECIM PM 2.0 onwards.

6.3.2 Content of ROP Files

The PM report file format is according to 3GPP TS 32.432 type A and is built on the example in 3GPP TS 32.435 Annex A.2. COM also supports multivalue counters, which are not part of the 3GPP specification.

COM PM supports job grouping. This is indicated by attribute `jobGroupingSupport` in `PmMeasurementCapabilities`. The attribute is set to `true` during COM PM startup. ROP files are written as follows:

- If attribute `jobGroup` in the `PmJob` MO is present, a separate report file is generated containing all measurements for each `PmJob` containing this `jobGroup`. The `<UniqueId>` part of the filename is appended with `"_<jobGroup>"`.



- If attribute `jobGroup` in the `PmJob` MO is absent, the default behavior is to include the measurement results for all such jobs in a single PM report file per GP, containing the results of those jobs for which no `jobGroup` tag was assigned.

If a `MeasurementType` instance (counter) is measured in more than one PM Job, the measurement result is reported separately for each job in a measurement report file.

If COM does not support ECIM PM 2.1, attribute `jobGroupingSupport` is not set and all measurement results are written to a single PM report file per GP.

6.3.3 Job Reporting Period

COM does not support measurements from several GPs in one ROP file. Attributes `reportingPeriod` and `granularityPeriod` of the `PmJob` MO are used to write the report. COM PM expects these attributes to have the same value. The validation of these attributes is expected to be done at job creation time by the PM service in the SA/MW. COM PM sets attribute `fileRPSupported=false` in the `PmMeasurementCapabilities` class during startup. This signals to the SA/MW that COM PM does not support different values to `reportingPeriod` and `granularityPeriod` for each individual `PmJob` when `fileRPSupported=false` and the SA/MW can validate this

Example

The collection of measurement results is started when the time is 0:00. Six jobs are set up, two (job 1 and 2) have a GP of one minute, two (job 3 and 4) have a GP of five minutes, and two (job 5 and 6) have a GP of 15 minutes. The creation of the files is as shown in Table 8.

Table 8 Example of Created Files

Time	Number of Created Files	Content
Between 0:00 and 0:01	0	No files are created. Measurement time collection has started at 0:00 and the GP has not ended.
Between 0:01 and 0:02	1	Measurement results for jobs 1 and 2 during period (0–1).
Between 0:02 and 0:03	1	Measurement results for jobs 1 and 2 during period (1–2).
Between 0:03 and 0:04	1	Measurement results for jobs 1 and 2 during period (2–3).



Time	Number of Created Files	Content
Between 0:04 and 0:05	1	Measurement results for jobs 1 and 2 during period (3–4).
Between 0:05 and 0:06	2	Measurement results for jobs 1 and 2 during period (4–5).
		Measurement results for jobs 3 and 4 during period (0–5).
Between 0:06 and 0:07	1	Measurement results for jobs 1 and 2 during period (5–6).
Between 0:10 and 0:11	2	Measurement results for jobs 1 and 2 during period (9–10).
		Measurement results for jobs 3 and 4 during period (5–10).
Between 0:14 and 0:15	1	Measurement results for jobs 1 and 2 during period (13–14).
Between 0:15 and 0:16	3	Measurement results for jobs 1 and 2 during period (14–15).
		Measurement results for jobs 3 and 4 during period (10–15).
		Measurement results for jobs 5 and 6 during period (0–15).

One minute is the upper time bound that it takes to get the results for a job written into a file after the end of the GP. After this time, the absence of an expected report file or the absence of measurement data for a configured active job is to be considered as “suspect”. This situation can be because of failures in the PM service in the SA/MW or because of file operation errors, in which case the COM PM functionality logs an error.

6.4 ECIM PM Model Compliance

COM does not deliver the PM model. However, the ownership of the fragment is split between COM and the SA/MW. The ownership of COM lies in the ROP file-related attributes of the `PmMeasurementCapabilities` class. For more information, refer to *COM ECIM Statement of Compliance*.