

Managed Object Model User Guide

USER GUIDE

Copyright

© Ericsson AB 2014–2016. 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
1.1	Target Groups	1
2	Managed Object Model	3
2.1	MOM Concepts	3
2.2	MOM Overview	3
3	MIB	5
3.1	Managed Objects	5
3.2	Managed Object Creation and Deletion	5
3.3	Managed Object Naming	5
3.4	Setting Attributes	6
3.5	Attribute Data Types	6
3.6	Calling Actions	6
3.7	Relationships between Managed Objects	7
3.8	Model Element Life Cycle	7
4	Presentation of MOM	9
4.1	Classes	9
4.2	Attribute Properties	12
4.3	Common Attributes	15
4.4	Deprecated Elements in MOM	16





1 Introduction

This document describes the Ericsson Managed Object Model (MOM), how to interpret its presentation within Customer Product Information (CPI), and how to use the MOM when interacting with a Management Information Base (MIB) on a Managed Element (ME), or when visible on an Operations Support System (OSS).

1.1 Target Groups

This document is aimed at end-user personnel involved in the Operation and Maintenance (O&M) of an ME.





2 Managed Object Model

This section describes the MOM concepts and provides an overview of the MOM.

2.1 MOM Concepts

A MOM is a structured collection of configuration information that defines the O&M capability on an ME. The MOM is defined as a set of classes, containing attributes representing the configuration, and actions representing the operations that can be started by the user. Using a MOM allows for harmonization of the O&M interfaces to Ericsson network equipment. Using a MOM also enables consistent product behavior across those interfaces, owing to the benefits of model driven software development.

The MOM is a static blueprint for the creation of the actual object model. A browsable HTML version of the MOM is delivered as part of the CPI for the ME.

The Managed Object Classes (MOCs) defined in the MOM are instantiated with real data on a deployed ME. A MOC becomes Managed Objects (MOs) also known as MO instances, holding configuration and state data. The data is used by the running system and stored in a database. The collection of MOs is called the MIB, which controls the configuration of an ME and its O&M functionality.

The MOs are monitored and manipulated by a management system or by a user through the provided interfaces. The read-only attributes in the MOs describe configuration state and operational values. The writable attributes control the operation and configuration for the particular network resource. The model does not define how an MO or network resource is implemented, only what can be seen in the interface.

This MO approach allows a single common representation to be used by different parts of the ME that offer O&M functionality, reducing software errors and inconsistencies.

2.2 MOM Overview

The Ericsson standard model for O&M has a single root element object called *ManagedElement*, which is the starting point for navigation. As shown in Figure 1, the model is organized into the following branches as first level:

- **SystemFunctions**
- **Transport**
- **Equipment**

- One or more function-specific branches named according to the ME function itself

Support functions are found on this first level. These functions are used by more than one application or component. These functions are not logical traffic functions, and are not part of these branches.

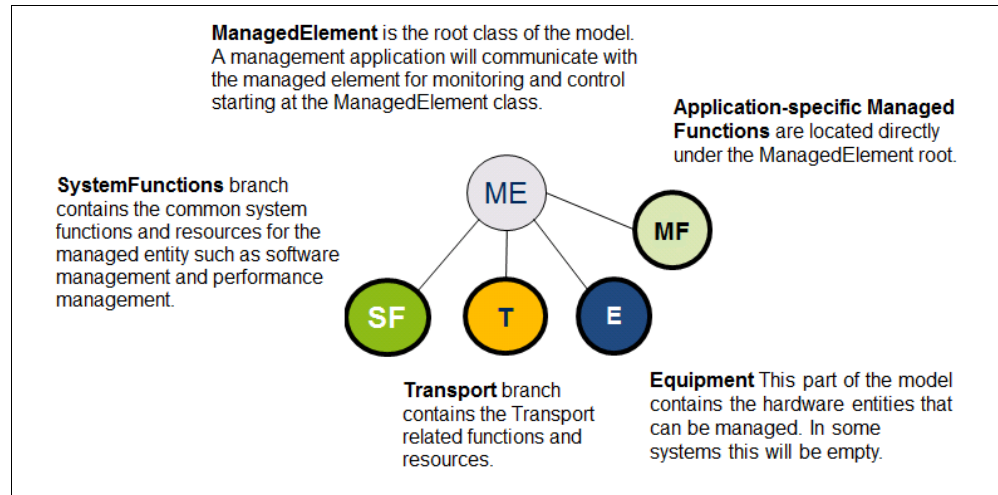


Figure 1 MOM Top-Level Structure



3 MIB

The MIB is the set of MOs used by the running system on a given ME. Managing the ME using the O&M interface is achieved by interaction with the instance model. The following operations can be performed:

- Create and delete MOs
- Set and read the MOs and the data they hold
- Start operations provided by the MOs

This interaction can be done directly through the Ericsson Command-Line Interface (ECLI), or indirectly using Ericsson OSS software, such as the NETCONF interface.

3.1 Managed Objects

In this O&M context, MOs are entities that are accessed through the O&M interface and represent a network resource for management. The MO is an instance of a MOC as defined in a MOM.

3.2 Managed Object Creation and Deletion

The primary interaction with the overall instance model is the reading, writing, creation, and deletion of MOs. Often, the MOs are created automatically by the ME itself. Depending on the capability, certain MOs can be created and deleted by the user or OSS management systems. MOs are created within the hierarchy, that is, the parent MO for an MO must exist before the MO is created.

3.3 Managed Object Naming

All MOs have a key identifier attribute. It allows addressing of each MO using a common hierarchical single-root scheme, based on industry standard 3GPP® Distinguished Names (DNs). The unique addressing of each MO and of each attribute within the MOs is supported. MOCs generally use key attributes with format *<MOC name with lowercase first character>Id*, for example, *fmAlarmId* in MOC *FmAlarm*.

The value of the key identifier attribute is used to form the Relative Distinguished Name (RDN) for that MO, for example, *fmAlarmId=alarm4*. The overall DN for the MO is formed by the sequence of RDNs starting from the root object of the model. The DN that starts at MO *ManagedElement* on the ME is a Local Distinguished Name (LDN). A Full Distinguished Name (FDN), when presented on a management system, starts at the management system own root object.

Example of an LDN: An MO *FmAlarm* has key attribute *fmAlarmId* with value *alarm4*. If the values of the parent object key identifiers are as follows, then



the DN of the MO is `ManagedElement=1, SystemFunctions=1, Fm=1, FmAlarmModel=SWM, FmAlarm=alarm4`.

Note: The 3GPP standard 32.300 details how the use of `Id` in the name of the key attribute allows DNs to omit the key attribute name.

3.4 Setting Attributes

The value of writable attributes in MOs can be set directly using a supported interface such as the ECLI, or through a management system. The ECLI enforces data types, properties, and constraints specified in the MOM to ensure that invalid user entries are prevented.

3.5 Attribute Data Types

The basic data types available for attributes are as follows:

boolean	"true" or "false"
enumeration	An aggregated type used to enumerate a fixed set of literals
integer	Signed and unsigned integer types
string	Alphanumeric characters
string (key)	For key attributes, the only allowed characters for the value, as specified by 3GPP standard 32.300 version 7.2.0, are "A–Z", "a–z", "0–9", "-", "_", "/", ".", "%", "&", "!", "?", " " (space), and ":".
	Note: The implementation does not enforce the use of these characters, but the results from the use of other characters cannot be guaranteed.
struct	A compound data type that groups element members, which can be of different types

Data types can also be defined within the MOM.

3.6 Calling Actions

Actions or operations provided by the O&M interface are similarly callable using the supported interfaces. An action is started in the context of the MO instance to tell the ME to perform a task related to the specific MO. Action start control, error reporting, and other features depend on the interface.

An action can have one or more input parameters and a return value. All parameters are defined as input to the action, and are not used to output



values. Any parameters in the action without default values must have values provided when the action is called. The result from executing an action is described in its MOM documentation. The execution can affect more than one MO instance and its read-only attributes.

3.7 Relationships between Managed Objects

MOs are connected by relationships to allow navigation from one MO to another. The relationships can be parent-child containment relationships, or associations between different model elements. Combined in the overall deployed MOM, these relationships produce the MO hierarchy tree, including the specified inter-MO relationships.

The MOM also includes the property of parent-child cardinality, which provides predefined constraints on how many instances of one MO can be connected to another MO. The cardinality can be a range that the implementation satisfies, or a number.

3.8 Model Element Life Cycle

Each model element contains a property that describes its life cycle state, to provide information that helps to keep compatibility between changing models and receiving clients. The states that can be presented for an element are as follows:

Current	The normal state, which means that the definition is valid and functional.
Deprecated	The element is marked for future deletion, typically when replaced by a new modified element or solution in parallel. It provides a window of compatibility to receiving client applications during which they can adapt to the new model.
Obsolete	The element has changed behavior compared to when it was <code>Current</code> and will be deleted in the future. The element can still be accessed, but the results depend on the product, are not specified, and can include no support. <code>Obsolete</code> is a variant of <code>Deprecated</code> . Both states are formally backwards compatible, but only <code>Deprecated</code> is functionally backwards compatible. The state <code>Obsolete</code> is normally not used, only when all other options are not feasible.





4 Presentation of MOM

A browsable MOM is provided in HTML as part of CPI.

In the library, the MOM is found under **Managed Object Management**, see Figure 2. In the tree structure, you can select the type of information to browse, such as **Classes**, **Attributes**, **Actions**, **Structures**, **Enumerations**, **Derived Data Types**, and **Instances**, depending on the MOM contents.

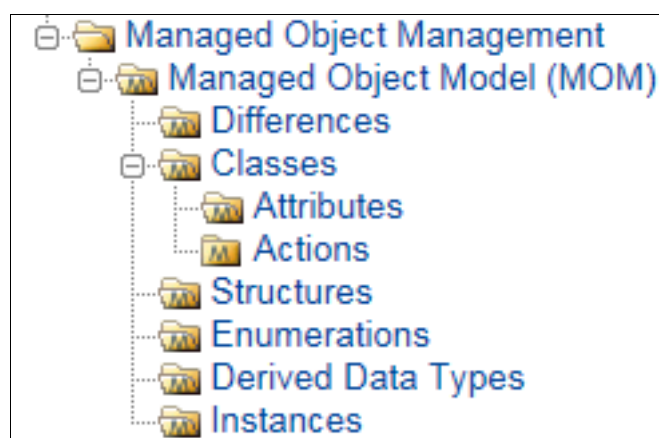


Figure 2 MOM Structure in the Library

The type of information to browse can also be selected on the front page of the MOM, see Figure 3. Attributes and actions are found under **Classes**. The differences in the MOM since the last release can also be included, and are in that case found under **Differences**.

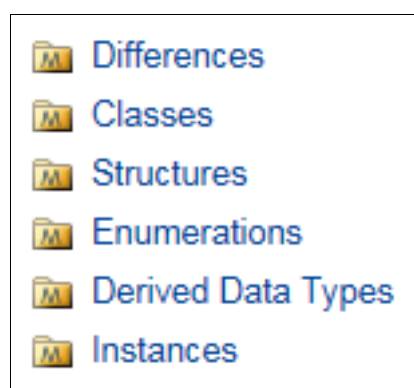


Figure 3 MOM Structure

4.1 Classes

MO classes in the MOM are presented with their place in the hierarchy, a description, and a list of attributes and actions. The part of the overall hierarchy

is shown, with the selected class in bold, see Figure 4. The parent classes it is placed under and the child classes that can exist under it are displayed with indentations.

If a class cannot be created or deleted by a user or a management system, the text **This MO is created by the system** is displayed.

After this information, the actions (if any) and attributes for this class are displayed.

class BrmBackupManager

ManagedElement

+--SystemFunctions

+--BrM

+--**BrmBackupManager**

+--BrmBackup [0..]

+--BrmBackupHousekeeping [0..1]

+--BrmBackupLabelStore [0..1]

+--BrmBackupScheduler [0..1]

Serves as container for Backup instances of a particular backupType and backupDomain.

This class also provides the actions for creating new backups, importing backups from other locations, and deleting existing backups of the corresponding type and domain.

This MO is created by the system.

Actions

boolean	<div>cancelCurrentAction ();</div> <div>Cancel an ongoing asynchronous createBackup, deleteBackup, or importBackup operation.</div>
boolean	<div>createBackup (string name);</div> <div>Creates a new backup of the backupType and backupDomain managed by the BackupManager. This is an asynchronous action - progress and result will be reported in the ReportProgress attribute.</div>

Figure 4 MO Class

4.1.1 Relationships between MO Classes

There are two relationships between MO classes, as follows:

- Containment, parent-child relation
- Association relation



4.1.2 Containment, Parent-Child Relation

In the MOM diagrams, parent-child relationships that define containment between MOs are indicated by UML aggregation notation, see Figure 5. In this example, MO-Y is the child of the parent MO-X. An instance of MO-Y cannot be created if no instance of MO-X exists. Neither can MO-X be deleted if it has any children, unless these are system-created.



Figure 5 Parent-Child Containment Relationship in MOM Diagrams

4.1.3 Association Relation

In the MOM diagrams, associations are indicated with solid lines, see Figure 6. It is a uni-directional association when MO-Y uses or starts methods in MO-X, but MO-X does not have to exist when MO-Y is created. The association is defined by a reference attribute in MO-Y. It is a bidirectional association when MO-X also contains a reserving attribute that references the MO with the association to it.

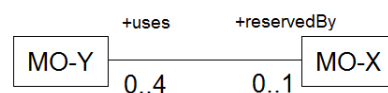


Figure 6 Bidirectional Association Relationship in MOM Diagrams

4.1.4 Cardinality of MO Relationships

Cardinality constraints between MOs are displayed in MOM diagrams by numerical indications at each end of the relationship.

In Figure 5, the notation 0..4 means that there can exist 0, 1, 2, 3, or 4 instances of MO-Y under MO-X only. Cardinality 1 on the MO-X side means that MO-X must exist before MO-Y can be created. Similarly for associations, the cardinality at each endpoint defines the number of targets that can be pointed at.



4.2 Attribute Properties

The following attribute properties are displayed in the MOM:

<data type>	The data type of the attribute can be a standard type or defined within the MOM.
key	Identifies the attribute that is the key (naming) attribute, used when creating the DN (address) for this MO.
mandatory	A value must be provided at the creation of the MO. Applies to read-write attributes with no default value with a minimum multiplicity of one or more.
noNotification	Attribute value change notifications are not sent to the management system for this attribute.
readOnly	The attribute is read-only over the O&M interfaces.
restricted	The attribute value can only be set once.

The following attribute properties are indicated by the absence of a tag in the display of the definition:

- If property `noNotification` is not displayed in the first column of the attributes table, the attribute sends a notification when changed.
- If property `mandatory` is not displayed, the attribute is optional to supply a value for at MO creation. It can have a default value that is set automatically when no value has been supplied by the user or management system.
- If property `readOnly` is not displayed, the attribute value can be changed over the O&M interface and the system provides updates internally.

4.2.1 Attribute Default Properties

If an attribute has a default value defined, it is displayed using the equal sign, see Figure 7.

Attributes	
<code>BrmManualBackupAutoDelete</code>	<code>autoDelete = ENABLED</code> Determines whether automated housekeeping of manual backups is enabled.

Figure 7 Attribute with Default Value

4.2.2 Derived Data Types

Derived data types are existing data types enhanced with extra restrictions and properties. Derived string data types contain, for example, length and content constraints. Derived integer data types contain extra range constraints. To

navigate to the definition of the derived data type, click the data type link in the attribute description.

4.2.3 Derived Integers with Range

An attribute of type derived integer with a range of allowed values presents the range lower and upper values in braces, see Figure 8.

int16	DayOfMonth { 0..31 }
-------	----------------------

Figure 8 Derived Data Type with Value Range

4.2.4 Strings with Pattern

An attribute of type derived string with defined length and restricted contents shows the length range in square brackets. The contents is shown as a POSIX[®] extended regular expression pattern with the text `Valid values`, see Figure 9. The description details the allowed value space so that it can be understood by a human.

<div> <div>derivedDataType Date</div> <div> <p>Represents the International Standard for the representation of date (ISO 8601) .</p> <p>The string format is "YYYY-MM-DD" (excluding quotes).</p> <p>The following pattern describes the Date format in detail: YYYY-MM-DD Where: YYYY = four digit year MM = two digit month (01=January, etc.) DD = two digit day of month (01 through 31)</p> </div> </div>	
string	<div> <div>Date</div> <div>Valid values: <code>^([0-9]{4})-(1[0-2] 0[1-9])-(0[1-9] [1 2][0-9] 3[0-1])\$</code></div> </div>

Figure 9 Derived Data Type

4.2.5 Enumerated Data Types

Enumerations are lists of integer-name pairs defining a fixed set of named values for an attribute, return value, or action parameter.

A specific attribute of enumerated type has its data type shown in the normal way in the attribute description column. In the example in Figure 10, the data type of attribute `certificateState` is the enumerated type `CertificateState`.



<code>CertificateState</code> <code>readOnly</code>	<code>certificateState</code> The current state of the certificate.
--	--

Figure 10 Attribute with Enumerated Data Type

To select the data type, click it. Enumerations are displayed as a list of possible integer values for an attribute, the associated defined name strings, and the value descriptions, see Figure 11.

enum CertificateState	
Certificate states.	
References from: <code>NodeCredential</code> ; <code>TrustedCertificate</code> ; <code>VendorCredential</code> ;	
1	NOT_VALID_YET Based on the <code>validFrom</code> date, the certificate is not yet valid.
0	VALID Based on the <code>validFrom</code> and <code>validTo</code> dates, the certificate is valid.
2	EXPIRED Based on the <code>validTo</code> date having passed, the certificate expired.
3	REVOKED The certificate was revoked by a trusted CA.

Figure 11 Enumerated Data Type

4.2.6

Structure Data Types

A struct type is a complex data type containing a set of members, each of which is individually typed. An example of the presentation of a struct is shown in Figure 12.

struct LmCapacityValue	
Defines a capacity value.	
References from: <code>CapacityKey</code> ;	
boolean	noLimit = false True if the value is unlimited, false if the value is defined by attribute "value".
int32	value The capacity value. The value is invalid if "noLimit=true".

Figure 12 Struct Data Type



4.2.7 Actions

Actions are displayed with the return type in the left-hand column. The action name is shown in the right-hand column, with input parameters and their data types in parentheses, see Figure 13.

boolean	removeUpgradePackage (UpgradePackage upgradePackage); Removes an UpgradePackage. This action removes the UpgradePackage MO specified as action parameter as well as all files temporarily stored in the ME and associated with the UP. The action returns immediately after invocation. The progress of the action can be tracked via the progressReport attribute. This action can be invoked when there is no other action in progress on this MO. The action returns false if the action could not start for any reason (e.g., another parallel action is in progress), otherwise returns true. Parameters Name: upgradePackage Description: An UpgradePackage MO to be removed.
----------------	--

Figure 13 Action with Return Type "boolean"

A return value of "boolean" indicates whether the operation triggered by the action was successfully terminated.

4.3 Common Attributes

The following common attributes are included in more than one MO:

operationalState

Specifies whether the MO is working (ENABLED) or failed (DISABLED). These values are set automatically.

administrativeState

Indicates whether the MO is administratively permitted to be used (UNLOCKED) or not permitted to be used (LOCKED).

The enumeration AvailStatus contains details about attribute operationalState and indicates why it has changed its value to DISABLED. The possible values are as follows:

- 0 (IN_TEST)
- 1 (FAILED)
- 2 (POWER_OFF)
- 3 (OFF_LINE)



- 4 (OFF_DUTY)
- 5 (DEPENDENCY)
- 6 (DEGRADED)
- 7 (NOT_INSTALLED)
- 8 (LOG_FULL)
- 9 (DEPENDENCY_LOCKED)
- 10 (DEPENDENCY_FAILED)
- 11 (DEPENDENCY_SHUTTINGDOWN)

4.4 Deprecated Elements in MOM

All elements are supported unless described otherwise. An element that is deprecated has the term *Deprecated* in the first line of the element description, similarly for state *Obsolete*. For details, see Section 3.8 Model Element Life Cycle on page 7.