INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.827.1

(10/2004)

SERIES Q: SWITCHING AND SIGNALLING

Q3 interface

Requirements and analysis for the common management functions of NMS-EMS interfaces

ITU-T Recommendation Q.827.1

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4, 5, 6, R1 AND R2	Q.120–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.799
Q3 INTERFACE	**Q.800–Q.849**
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1699
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000	Q.1700–Q.1799
SPECIFICATIONS OF SIGNALLING RELATED TO BEARER INDEPENDENT CALL CONTROL (BICC)	Q.1900–Q.1999
BROADBAND ISDN	Q.2000–Q.2999

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation Q.827.1

Requirements and analysis for the common management functions of NMS-EMS interfaces

Summary

This Recommendation provides the requirements and analysis for the common management functions for NMS-EMS interface which can be used for network technology-specific management interfaces. The requirements and analysis for the management interface between EMS and NMS are provided using the TMN interface specification methodology described in ITU-T Rec. M.3020.

Source

ITU-T Recommendation Q.827.1 was approved on 7 October 2004 by ITU-T Study Group 4 (2001-2004) under the ITU-T Recommendation A.8 procedure.

Keywords

Common management, Unified Modelling Language (UML).

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

ITU-T Recommendation Q.827.1

Requirements and analysis for the common management functions of NMS-EMS interfaces

1 Scope

This Recommendation provides the requirements and analysis of common management functions applicable at NMS-EMS interface. The term "common" is used in two different contexts. In the first context, it includes analysis of the requirements that are applicable for fault, performance, configuration and security management functions. The second usage is to analyse the requirements that are independent of the specific technology. An example of the first context is logging events for alarms, performance threshold crossing, and inventory changes to the data of the entities. An example of the second context is the method for collecting and reporting performance measurements independent of the network technologies, such as EPON or SDH-DLC.

This Recommendation follows the interface specification methodology described in ITU-T Rec. M.3020 [2]. In this Recommendation, the Element Management System (EMS) is an Operation System (OS) [1] used to manage the individual network elements supporting specific network technologies as well as the networks between them. One or more systems may be required depending on the different supplier products and geographic distribution of the elements in the network. The Network Management System (NMS) represents an integrated management OS across different technology and supplier systems.

The management functions covered in this Recommendation include common management, performance management, fault management and security management. Configuration management and performance parameters are network technology-dependent, and they will be addressed in other technology-specific Recommendations. Security management is for future study. These management functions are described and divided by UML use cases in the requirements. The analysis provides the managed entities that support a protocol-neutral information model for common management functions, and also explains the static and dynamic relationships between these managed entities using UML class diagrams and sequence diagrams.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[1] ITU-T Recommendation M.3010 (2000), *Principles for a telecommunication management network*.

[2] ITU-T Recommendation M.3020 (2000), *TMN interface specification methodology*.

[3] ITU-T Recommendation M.3100 (1995), *Generic network information model*.

[4] ITU-T Recommendation M.3200 (1997), *TMN management services and telecommunications managed areas: overview*.

[5] ITU-T Recommendation M.3400 (2000), *TMN management functions*.

[6] ITU-T Recommendation X.720 (1992) | ISO/IEC 10165-1:1993, *Information technology – Open Systems Interconnection – Structure of management information: Management information model*.

[7] ITU-T Recommendation X.721 (1992) | ISO/IEC 10165-2:1992, *Information technology – Open Systems Interconnection – Structure of management information: Definition of management information*.

[8] ITU-T Recommendation X.731 (1992) | ISO/IEC 10164-2:1993, *Information technology – Open Systems Interconnection – Systems Management: State management function*.

[9] ITU-T Recommendation X.733 (1992) | ISO/IEC 10164-4:1992, *Information technology – Open Systems Interconnection – Systems Management: Alarm reporting function*.

[10] ITU-T Recommendation X.734 (1992) | ISO/IEC 10164-5:1993, *Information technology – Open Systems Interconnection – Systems Management: Event report management function*.

[11] ITU-T Recommendation X.735 (1992) | ISO/IEC 10164-6:1993, *Information technology – Open Systems Interconnection – Systems Management: Log control function*.

[12] ITU-T Recommendation X.739 (1993) | ISO/IEC 10164-11:1994, *Information technology – Open Systems Interconnection – Systems Management: Metric objects and attributes*.

[13] ITU-T Recommendation X.792 (1999), *Configuration audit support function for ITU-T applications*.

3 Terms and definitions

No new definitions are needed in this Recommendation.

4 Abbreviations

This Recommendation uses the following abbreviations:

ASAP	Alarm Severity Assignment Profile
CM	Configuration Management
CMIP	Common Management Information Protocol
CORBA	Common Object Request Broker Architecture
EFD	Event Forwarding Discriminator
EMS	Element Management System
EPON	Ethernet Passive Optical Network
FM	Fault Management
FS	Function Set
FTP	File Transfer Protocol
ID	Identifier
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
MO	Managed Object
NMS	Network Management System
OS	Operations System
PM	Performance Management

QoS Quality of Service

SDH-DLC Synchronous Digital Hierarchy – Digital Loop Carrier

TMN Telecommunications Management Network

UML Unified Modelling Language

5 General overview

The EMS shown in Figure 5-1 is used to manage the individual network elements supporting some specific network technologies. One or more systems may be required depending on the different supplier products and geographic distribution of the elements in the network. NMS, the network layer management system shown in Figure 5-1, represents an integrated management OS across the different technology and supplier systems. The logical representation shown by NMS may be realized by one or more physical interfaces. Figure 5-1 shows the Q interface addressed in this Recommendation.

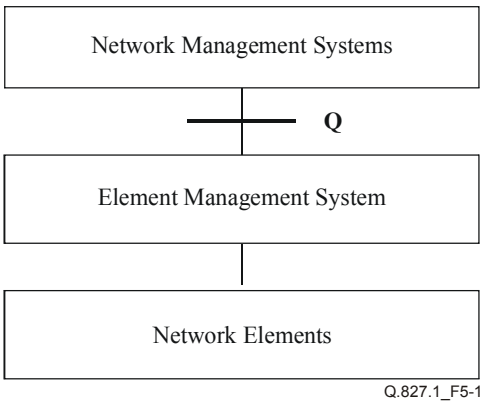

Figure 5-1/Q.827.1 – Reference interface

6 Requirements

6.1 Business level requirements

This Recommendation only focuses on the management interface between NMS and EMS and the interface management functions associated with them. Through the interface, NMS can query and modify configuration information, and EMS should report changes in configuration, performance data, and fault information to NMS.

6.1.1 Actor roles

The actor in this Recommendation is NMS. NMS is the network management system that manages the whole network system by interacting with suppliers' EMSs.

6.1.2 Telecommunications resources

Both the suppliers' EMSs and the managed network equipments are viewed as relevant telecommunications resources in this Recommendation.

6.1.3 High-level use-case diagrams

This clause contains the high-level UML use-case diagrams that summarize the functionalities and interfaces of EMS.

The first overview use-case diagram (Figure 6-1) shows the main management function sets (FS) involved in the NMS-EMS management interface.

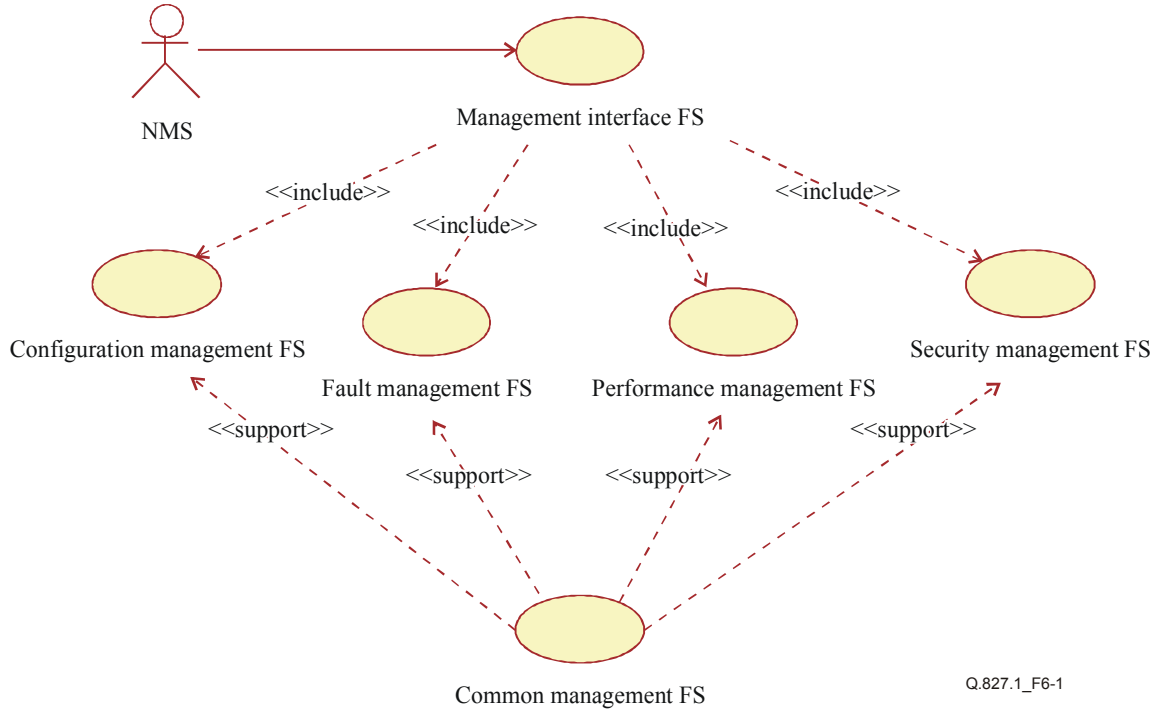

Figure 6-1/Q.827.1 – NMS-EMS management interface function set

The following four use-case diagrams (Figures 6-2 to 6-5) show the common management function sets, which are: Common management function set overview (Figure 6-2), Notification management function set (Figure 6-3), Log management function set (Figure 6-4), and Bulk data transfer control function set (Figure 6-5).

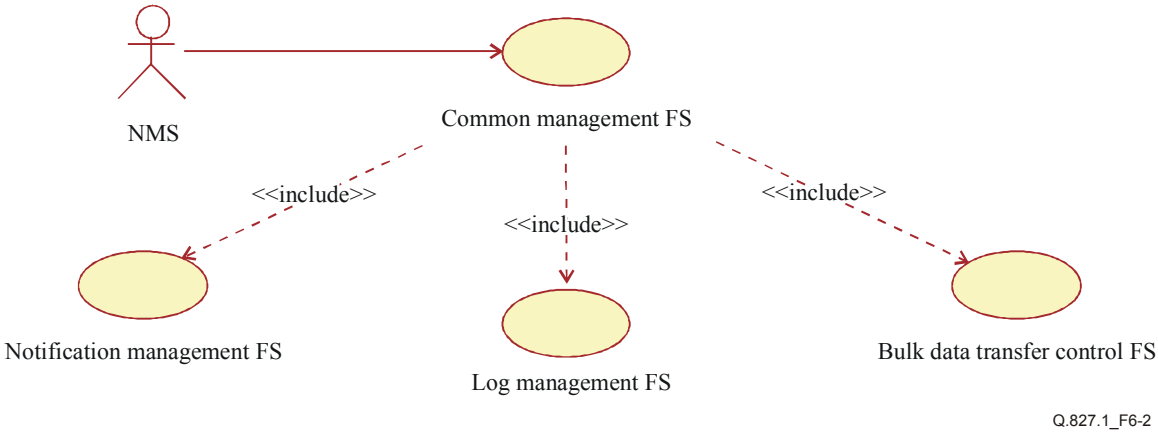

Figure 6-2/Q.827.1 – Common management function set overview

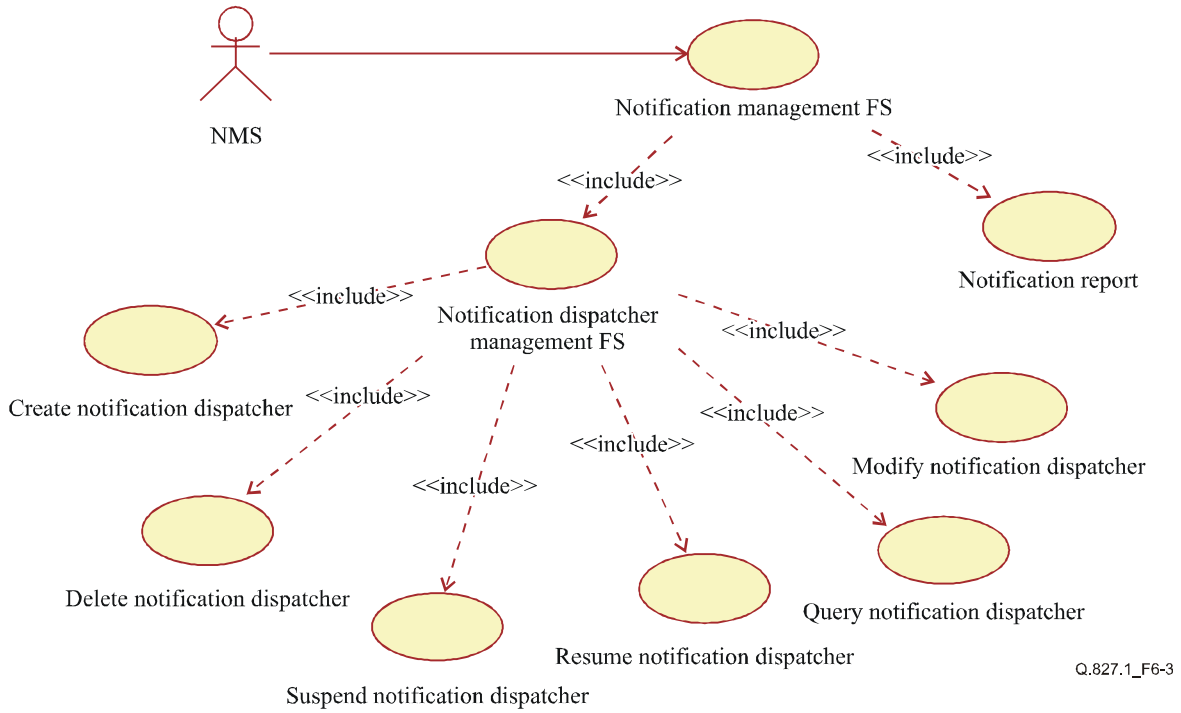

Figure 6-3/Q.827.1 – Notification management function set

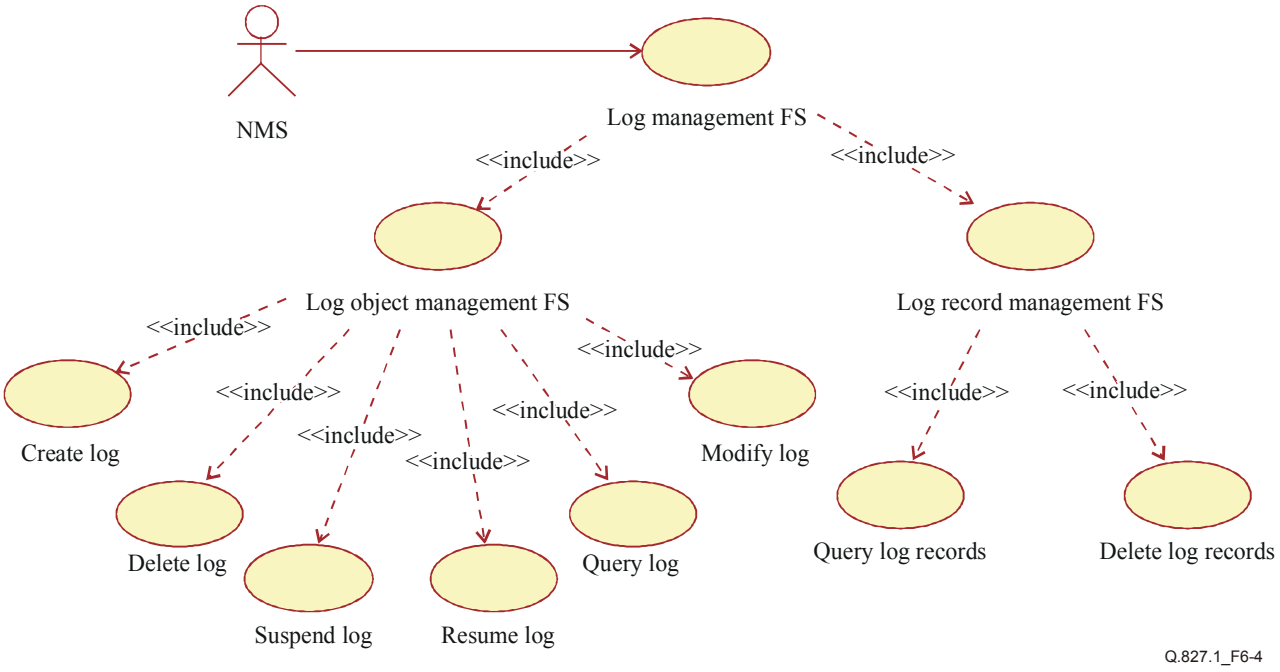

Figure 6-4/Q.827.1 – Log management function set

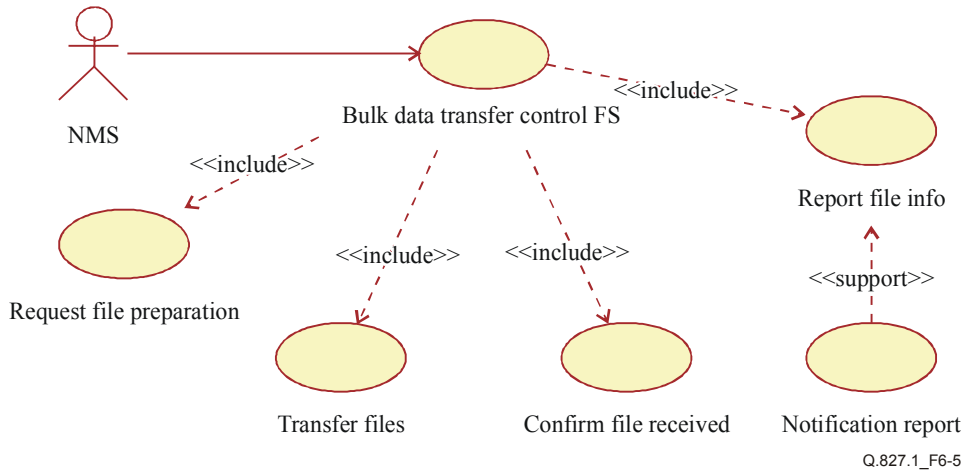

Figure 6-5/Q.827.1 – Bulk data transfer control function set

Configuration management function set is network technology dependent, so it will not be addressed in this Recommendation.

The following three diagrams show the functions involved in the performance management function set. Figure 6-6 illustrates performance management function set, Figure 6-7 is the performance measurement management function set use-case diagram, and Figure 6-8 is the threshold management function set use-case diagram.

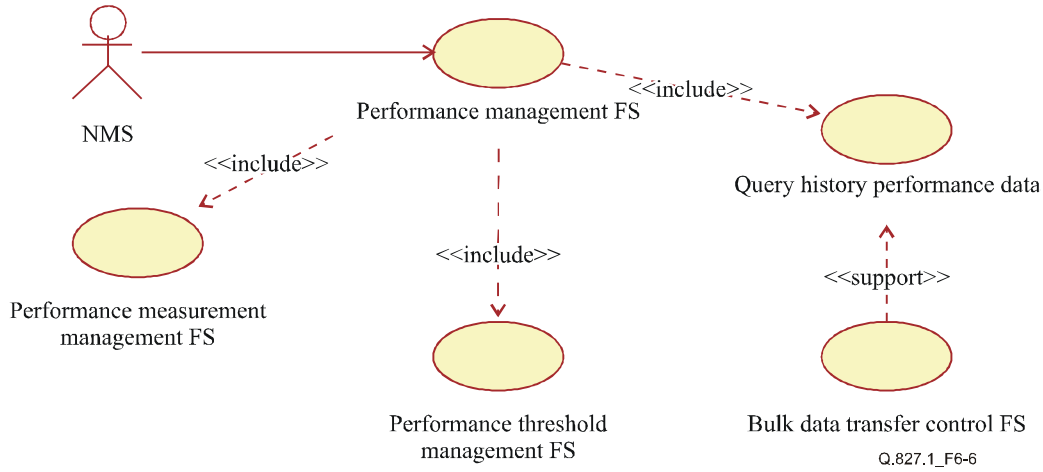

Figure 6-6/Q.827.1 – Performance management function set

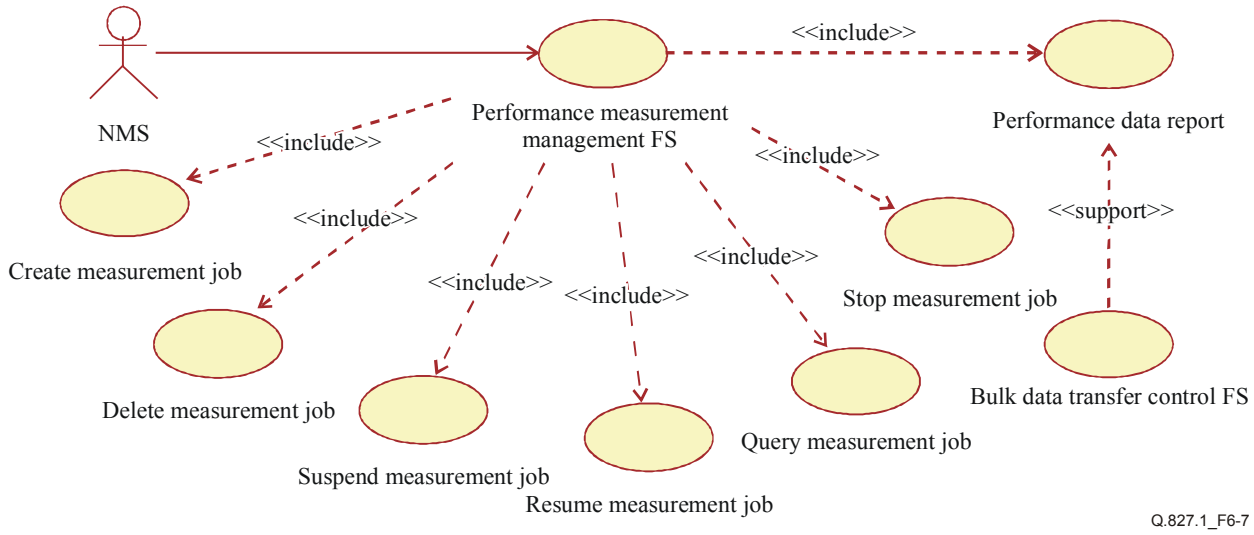

Figure 6-7/Q.827.1 – Performance measurement management function set

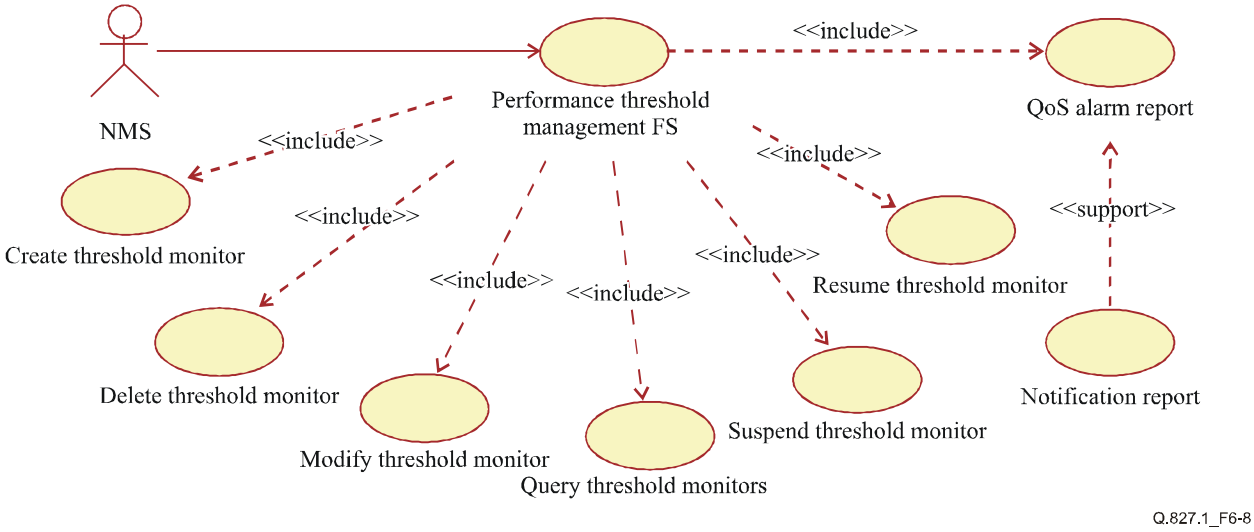

Figure 6-8/Q.827.1 – Performance threshold management function set

The following two use-case diagrams (Figures 6-9 and 6-10) show the functions involved in fault management function set.

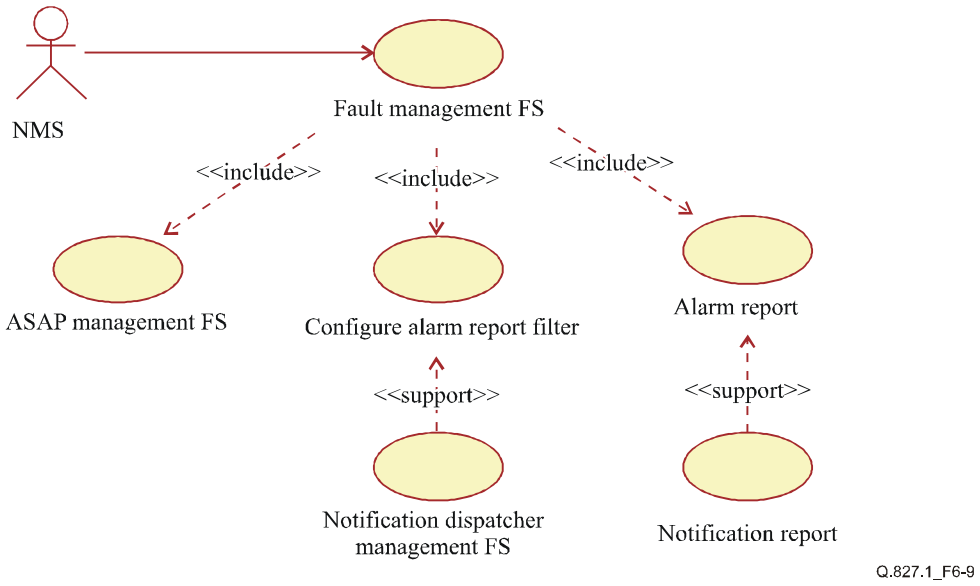

Figure 6-9/Q.827.1 – Fault management function set

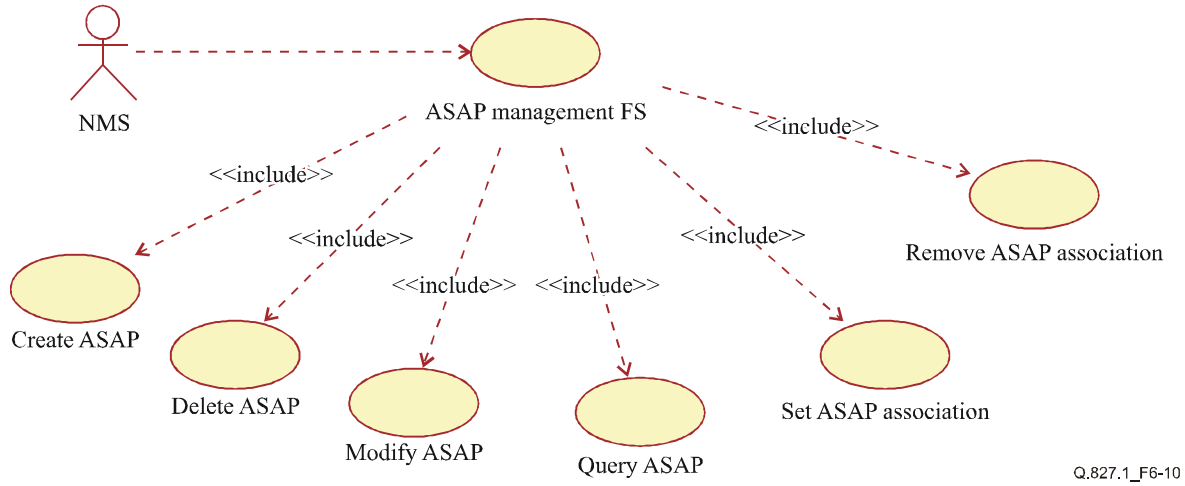

Figure 6-10/Q.827.1 – ASAP management function set

6.2 Specification level requirements

This clause contains textual details for each of the use cases shown in the high-level use-case diagrams of the previous clause. The details are provided to clarify the roles of external actors and telecommunications resources, to establish the basis for interactive diagrams in the analysis part, and to refine the previous high-level use-case diagrams to a specification level. Use-case details include the following components:

- **Summary**: short summary of use-case functionality referencing TMN functionality as needed.
- **Assumptions**: listing of requirements surrounding the use case that would affect the design of the application code of a supplier's EMS.
- **Actors**: actors which are described in 6.1.1.
- **Preconditions**: identifies the trigger for the use-case commencement.
- **Description**: detailed textual rendition of the functionality of the use case, including stops where exceptions can occur.

- **Exceptions**: identifies unsuccessful completion circumstances for the use case.
- **Post-conditions**: identifies the conditions that will hold if the use case ends successfully.

The use-case details are listed by the use-case title shown in the use-case diagrams of the previous clause.

6.2.1 Common management function set

6.2.1.1 Overview

The common management function set includes the notification management function set, log management function set and bulk data transfer control function set. Figure 6-2 shows the details.

The notification management function set can be divided into notification report function and notification dispatcher management function set. The latter also includes the following functions: create notification dispatcher, delete notification dispatcher, suspend notification dispatcher, resume notification dispatcher, modify notification dispatcher, and query notification dispatcher. Figure 6-3 shows the details.

The log management function set can also be divided into log object management function set, and log record management function set (see also ITU-T Rec. X.735 [11] for more information). Log object management FS includes create log, delete log, suspend log, resume log, modify log, and query log functions. Log record management FS includes query log records and delete log records functions. Figure 6-4 shows the details.

In the cases that the managed system is required to transfer bulk data, the bulk data transfer control function set will be used to improve the transport efficiency and reliability. This function set is involved in configuration management, fault management, performance management, and it may also be used in common management for querying log records.

The bulk data transfer control function set includes the following functions: request file preparation, transfer file, confirm file received and report file preparation information. Figure 6-5 shows the details.

The "Common management function set" here falls into the first context explained in clause 1 "Scope", including functions that are applicable for fault, performance, configuration and security management functions and can be achieved in the same mechanisms.

6.2.1.2 Notification management function set

6.2.1.2.1 Notification report

Summary: EMS can report notifications related to configuration, fault, performance or common management to NMS. See also ITU-T Rec. X.734 [10] for more descriptions.

Assumptions: The communication between EMS and NMS is available.

Actors: NMS.

Preconditions: An object has been created or deleted. An attribute value or state of an object has changed. Alarm occurs, or the bulk data file preparation is ready, or error occurs during bulk data file preparation.

Description: The information associated with configuration, fault and performance can be reported to NMS using the notification report function. The object creation notification, object deletion notification, attribute value change notification and state change notification are used in configuration management. Equipment alarm notification, environment alarm notification, communication alarm notification and processing error alarm notification are used in fault management. QoS alarm notification is used in performance management. Bulk data transfer ready and bulk data transfer preparation error notifications are used in common management.

Exceptions: Communication error.

Post-conditions: EMS will trigger notifications to the appropriate notification dispatcher(s), and those passing the corresponding filtering criteria will be forwarded to NMS.

6.2.1.2.2 Notification dispatcher management function set

6.2.1.2.2.1 Create notification dispatcher

Summary: NMS can send a request to EMS to create a notification dispatcher.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of creating a notification dispatcher.

Actors: NMS.

Preconditions: NMS needs to receive notifications from EMS.

Description: NMS can create a notification dispatcher through the management interface. In the case that EMS can initialize a notification dispatcher instance(s) by itself, it is not required to provide this function in the interface. (For example, the CORBA-based interface does not require the notification dispatcher creation function. As for the CMIP-based interface, this function is provided to create an EFD instance.) In this use case, NMS requests EMS to create a notification dispatcher. The parameters in the request include the destination(s), the initial administrative state and filtering criteria of the dispatcher.

If the dispatcher has been created successfully, EMS will then return the identifier of the notification dispatcher and may send a corresponding object creation notification to NMS. The subsequent notifications will be forwarded according to the attribute values of the dispatcher. If the creation fails, EMS will return error information. The possible errors are listed under "Exceptions".

Exceptions: Invalid parameter; EMS processing error; Communication error.

Post-conditions: The notification dispatcher is created by EMS at the request of NMS. EMS may send the corresponding object creation notification to NMS, and the newly created dispatcher acts on new notifications.

6.2.1.2.2.2 Delete notification dispatcher

Summary: NMS can delete a notification dispatcher through the management interface.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of deleting a notification dispatcher.

Actors: NMS.

Preconditions: NMS no longer needs to receive notifications from the specified notification dispatcher in EMS. The specified notification dispatcher is suspended.

Description: NMS can delete a notification dispatcher through the management interface. In the case that EMS can initialize a notification dispatcher instance(s) by itself, it is not required to provide this function in the interface (as in CORBA-based interfaces). In this use case, NMS requests EMS to delete a notification dispatcher. The parameter in the request is the ID of the notification dispatcher. If the dispatcher has been deleted successfully, EMS may then send a corresponding object deleted notification to NMS. If the deletion fails, EMS will return error information.

Exceptions: Unknown dispatcher; Dispatcher not suspended; EMS processing error; Communication error.

Post-conditions: The notification dispatcher is deleted by EMS on the request of NMS. EMS may send the corresponding object deletion notification to NMS. No more notifications will be forwarded to NMS by this deleted notification dispatcher.

6.2.1.2.2.3 Suspend notification dispatcher

Summary: NMS can suspend a notification dispatcher through the management interface. Thereafter, the dispatcher will not forward any notifications.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of suspending a notification dispatcher.

Actors: NMS.

Preconditions: The specified notification dispatcher exists in EMS and it is not suspended. NMS temporarily does not want to receive notifications from the dispatcher in EMS, or NMS needs to change the attribute values of the dispatcher.

Description: In this use case, NMS sends a request to EMS to suspend a notification dispatcher, that is, to set the administrative state of the dispatcher from "unlocked" to "locked". If the operation succeeds, the dispatcher will not forward any notifications until it is resumed. If the operation fails, EMS will return error information.

Exceptions: Dispatcher already suspended; EMS processing error; Communication error.

Post-conditions: The notification dispatcher is suspended by EMS on the request of NMS. EMS may send the corresponding state change notification to NMS, and the dispatcher does not forward notifications any more until it is resumed.

6.2.1.2.2.4 Resume notification dispatcher

Summary: NMS can resume a suspended notification dispatcher through the management interface, so that the dispatcher can forward notifications again.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of resuming a notification dispatcher.

Actors: NMS.

Preconditions: The specified notification dispatcher exists in EMS and it is suspended. NMS wants to receive notifications again from the specified dispatcher in EMS.

Description: In this use case, NMS sends a request to EMS to resume a suspended notification dispatcher, that is, to set the administrative state of the dispatcher object from "locked" to "unlocked". If the operation succeeds, the dispatcher will begin to forward notifications again. If the operation fails, EMS will return error information.

Exceptions: Dispatcher not suspended; EMS processing error; Communication error.

Post-conditions: The notification dispatcher is resumed by EMS on the request of NMS. EMS may send the corresponding state change notification to NMS, and the dispatcher acts on notifications again.

6.2.1.2.2.5 Modify notification dispatcher

Summary: NMS can modify the attribute values of a notification dispatcher through the management interface.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of modifying a notification dispatcher.

Actors: NMS.

Preconditions: NMS needs to change the filtering criteria of a notification dispatcher in EMS, or change the destination(s) of notifications to be forwarded by the dispatcher. The specified notification dispatcher exists in EMS and it is suspended.

Description: In this use case, NMS sends a request to EMS to modify a notification dispatcher. The filtering criteria and the destination(s) can be modified. If the modification succeeds, the dispatcher will forward notifications according to the new attribute values. If the modification fails, EMS will return error information.

Exceptions: Invalid parameter; Dispatcher not suspended; EMS processing error; Communication error.

Post-conditions: The notification dispatcher is modified by EMS on the request of NMS. EMS may send the corresponding attribute value change notification to NMS. The newly modified dispatcher then forwards notifications with the new filtering criteria to the new destination(s).

6.2.1.2.2.6 Query notification dispatcher

Summary: NMS can query the attribute values of a notification dispatcher in EMS.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of querying a notification dispatcher.

Actors: NMS.

Preconditions: NMS needs to query the attribute information of a notification dispatcher in EMS. The specified notification dispatcher exists in EMS.

Description: NMS can query the attribute values of a notification dispatcher such as administrative state, filtering criteria, destinations and so on. In this use case, NMS sends a request to query a notification dispatcher. The parameters in the request include the names of the attributes to be queried. If the function succeeds, EMS will return the specified attribute values. Otherwise, EMS will return error information.

Exceptions: Invalid parameter; EMS processing error; Communication error.

Post-conditions: EMS returns the attribute values requested by NMS.

6.2.1.3 Log management function set

6.2.1.3.1 Log object management function set

6.2.1.3.1.1 Create log

Summary: NMS can create a log through the management interface.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of creating a log.

Actors: NMS.

Preconditions: NMS needs EMS to log event information so that the information may be queried in the future for some specific situations such as data loss.

Description: NMS can create a log through the management interface. In the case that EMS can initialize a log instance(s) by itself, it is not required to provide this function in the interface. In this use case, NMS requests EMS to create a log. The parameters in the request include the initial administrative state, max log size, log full action, capacity alarm threshold, and filtering criteria of the log object.

If the log has been created successfully, EMS will then return the identifier of the log and send an object creation notification to NMS. Whether events will be recorded as log records depends on the filtering criteria defined in the log. If the creation fails, EMS will return error information.

Exceptions: Invalid parameter; EMS processing error; Communication error.

Post-conditions: The log is created by EMS on the request of NMS. EMS returns the identifier of the log and may send the corresponding object creation notification to NMS. The events will be recorded according to the filtering criteria in the newly created log.

6.2.1.3.1.2 Delete log

Summary: NMS can delete a log through the management interface.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of deleting a log.

Actors: NMS.

Preconditions: The specified log exists in EMS and it is suspended. NMS does not need the log in EMS to record any event information.

Description: NMS can delete a log through the management interface. In the case that EMS system can initialize a log instance(s), it is not required to provide this function in the interface. In this use case, NMS requests EMS to delete a log. The parameter in the request is the ID of the log object. If the log has been deleted successfully, EMS may then send the corresponding object deletion notification to NMS. If the deletion fails, EMS will return error information.

Exceptions: Unknown log, Log not suspended; EMS processing error; Communication error.

Post-conditions: The log and the associated log records are deleted by EMS on the request of NMS. EMS may send the corresponding object deletion notification to NMS.

6.2.1.3.1.3 Suspend log

Summary: NMS can suspend a log through the management interface. Thereafter, the log will not record any events until it is resumed.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of suspending a log.

Actors: NMS.

Preconditions: NMS temporarily needs to suspend a log in EMS. The specified log exists in EMS and it is not suspended. NMS temporarily does not want the log to record event information in EMS, or NMS needs to change the attribute values of the log object. The specified log exists in EMS.

Description: In this use case, NMS sends a request to EMS to suspend a log, that is, to set the administrative state of the Log object from "unlocked" to "locked". If the operation succeeds, the log object will not record any events until it is resumed. If the operation fails, EMS will return error information.

Exceptions: Log already suspended; EMS processing error; Communication error.

Post-conditions: The log is suspended by EMS on the request of NMS. EMS may send the corresponding state change notification to NMS, and the log does not record any events until it is resumed.

6.2.1.3.1.4 Resume log

Summary: NMS can resume a suspended log through the management interface, so that the log can record events again.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of resuming a log.

Actors: NMS.

Preconditions: The specified log exists in EMS and it is suspended. NMS wants the log to record events again in EMS.

Description: In this use case, NMS sends a request to EMS to resume a suspended log, that is, to set the administrative state of the log object from "locked" to "unlocked". If the operation succeeds, the log object will begin to record events again. If the operation fails, EMS will return error information.

Exceptions: Log not suspended; EMS processing error; Communication error.

Post-conditions: The log is resumed by EMS on the request of NMS. EMS may send the corresponding state change notification to NMS. The log continues to record events according to its filtering criteria.

6.2.1.3.1.5 Modify log

Summary: NMS can modify the attribute values of a log through the management interface.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of modifying a log.

Actors: NMS.

Preconditions: The specified log exists in EMS and it is suspended. NMS needs to change the attribute values of a log in EMS, such as the filtering criteria of events logging, the log full action, the capacity alarm threshold, and so on.

Description: In this use case, NMS sends a request to EMS to modify the attribute values of a log. The input parameters are the attribute names to be modified and the corresponding new values. The attributes that can be modified include: max log size, log full action, capacity alarm threshold, and filtering criteria. If the modification succeeds, the log will use new criteria to record events and act according to these new attribute values. If the modification fails, EMS will return error information.

Exceptions: Invalid parameter; Log not suspended; EMS processing error; Communication error.

Post-conditions: The log is modified by EMS on the request of NMS. EMS may send the corresponding attribute value change notification to NMS. The newly modified log then begins to record events with the new criteria and behaves according to the new attribute values.

6.2.1.3.1.6 Query log

Summary: NMS can query the attribute values of a log in EMS.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of querying a log.

Actors: NMS.

Preconditions: NMS needs to retrieve the attribute information of a log in EMS. The specified log exists in EMS.

Description: NMS can query the attribute values of a log, including administrative state, operational state, max log size, current log size, log full action, number of records, alarm status, capacity alarm threshold, filtering criteria, and so on. In this use case, NMS sends a request to query a log. The parameters in the request are the attribute names to be queried. If the operation succeeds, EMS will return the corresponding attribute values. Otherwise, EMS will return error information.

Exceptions: Invalid parameter; EMS processing error; Communication error.

Post-conditions: EMS returns the attribute values of the log requested by NMS.

6.2.1.3.2 Log record management function set

6.2.1.3.2.1 Query log records

Summary: NMS can query the log records contained in a log according to some criteria.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of querying log records.

Actors: NMS.

Preconditions: In the case that the notification information between NMS and EMS is not synchronized due to broken communication, abnormal data lost or other reasons, NMS needs to query log records in EMS to synchronize the notification information between two systems. The specified log exists in EMS.

Description: In this use case, NMS sends a "query log record" request to EMS. The parameters in the request are the filtering criteria and the time boundary. If the query succeeds, EMS will return the log records that satisfy the criteria and the time boundary. Otherwise, if the query fails, EMS will return error information.

Exceptions: Invalid parameter; No such log records; EMS processing error; Communication error.

Post-conditions: The corresponding log records are returned by EMS.

6.2.1.3.2.2 Delete log records

Summary: NMS can delete log records contained in a log according to certain criteria.

Assumptions: The communication between NMS and EMS is available. EMS supports the function of deleting log records.

Actors: NMS.

Preconditions: In the case where the information in some log records is of no use to NMS (too old or having been transferred to NMS), NMS can delete some or all log records in EMS. The specified log exists in EMS.

Description: In this use case, NMS sends a deletion request to EMS. The parameters in the request are the filtering criteria and the time boundary. If the deletion succeeds, the corresponding records that satisfy the criteria and the time boundary will be deleted. Otherwise, if the deletion fails, EMS will return error information.

Exceptions: Invalid parameter; No such log records; EMS processing error; Communication error.

Post-conditions: The log records that satisfy the filtering criteria and the time boundary are deleted.

6.2.1.4 Bulk data transfer control function set

6.2.1.4.1 Request file preparation

Summary: NMS can send a request to EMS to start a bulk data file preparation task.

Assumptions: The communication between NMS and EMS is available. EMS supports the FTP service.

Actors: NMS.

Preconditions: NMS needs to synchronize configuration audit data, performance history data, or log records information through the bulk data file transfer in EMS.

Description: NMS can request EMS to start a file preparation task using this function. On receiving this request, EMS will start to prepare the file(s) to be transferred. When EMS is ready for file transfer, it will send a "Bulk Data Transfer Ready" notification to NMS. If any exceptions occur during the preparation, EMS will send a "Bulk Data Transfer Preparation Error" notification to

NMS. In this use case, NMS sends a request to EMS to start a file preparation task. The parameters in the request include the file type, object selection (optional), job identifier (optional) and time boundary (optional). The file type may be one of configuration audit file, performance data file or log record file. For CM audit, and log record files, the object selection shall be specified to locate the managed object(s) for retrieving information from. For PM files, the identifier of the measurement job shall be specified. For PM and Log files, the time boundary can be applied. EMS then processes the request. If this operation succeeds, a file transfer ID will be returned to NMS for future uses. If the file has been prepared according to the request parameters, EMS will send a "Bulk Data Transfer Ready" notification to NMS. If the preparation fails, EMS will send a "Bulk Data Transfer Preparation Error" notification to NMS along with the possible causes.

Exceptions: Invalid parameter; EMS processing error; Communication error.

Post-conditions: EMS starts a file preparation task on the request of NMS. When the preparation is finished, EMS will send a "Bulk Data Transfer Ready" notification to NMS.

6.2.1.4.2 Transfer files

Summary: NMS retrieves the files from EMS.

Assumptions: The communication between NMS and EMS is available. EMS supports FTP service.

Actors: NMS.

Preconditions: NMS receives a "Bulk Data Transfer Ready" notification from EMS.

Description: Files can be transferred from EMS to NMS using FTP service. When NMS receives a "Bulk Data Transfer Ready" notification, it will begin to get the data files from EMS using FTP.

This function may actually be achieved by FTP lower level operations (such as FTP "get" operation), and no application level operation is needed for this function.

Exceptions: None (no application level exceptions).

Post-conditions: NMS gets the data files from EMS according to the file information specified in the received "Bulk Data Transfer Ready" notification. When file transfer finishes, NMS should check integrity and correctness of the received files.

6.2.1.4.3 Confirm file received

Summary: After data files have been transferred from EMS to NMS, NMS will inform EMS for confirmation.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: Data files for a transfer ID have been transferred from EMS to NMS.

Description: After the data files have been transferred from EMS to NMS, NMS will send a confirmation to EMS. The confirmation includes the file transfer ID and the status for each file. The status can be one of the following cases:

1) the specified file is successfully received and the syntax checking is OK;
2) the specified file is not found;
3) the specified file is received, but the file format is invalid; or
4) the specified file is received, but it is invalid for some other reasons.

When receiving this confirmation, EMS may then deal with the files without problems, for example, delete or move them to some other places. When the confirmation indicates there are some files not found or invalid, EMS should re-trigger the file preparation process for these files.

Exceptions: Unknown transfer ID; EMS processing error; Communication error.

Post-conditions: After receiving the confirmation, EMS can deal with the successfully confirmed files.

6.2.1.4.4 Report file information

Summary: The bulk data transfer control function set uses the notification report function described in the common management function set. The used notifications are the "Bulk Data Transfer Ready" notification and the "Bulk Data Transfer Preparation Error". When the data have been collected and the files have been prepared, EMS will send the "Bulk Data Transfer Ready" notification to NMS. When some exception occurs during the preparation, EMS will send the "Bulk Data Transfer Preparation Error" notification to NMS. Please refer to 6.2.1.2.1, "Notification Report" function in the Common management function set for details.

6.2.2 Configuration management function set

Configuration management function set is network technology dependent, and thus, it is outside the scope of this Recommendation.

6.2.3 Performance management function set

6.2.3.1 Overview

The performance management function set contains: Performance measurement management function set, Performance threshold management function set, and Query history performance data function, as depicted in Figure 6-6.

Performance measurement is the activity whereby the EMS periodically collects the performance data from physical equipments as well as logical entities and reports them to the NMS. Performance measurement management function set is provided to the NMS to manage the parameters related to performance measurement, through which performance data can be reported by the EMS according to the requests of the NMS.

Performance measurement management FS contains the following functions: Create measurement job, Delete measurement job, Stop measurement job, Suspend measurement job, Resume measurement job, Query measurement jobs, and Performance data report. Figure 6-7 shows the details.

(NOTE – The modify measurement job function can actually be achieved by deleting the existing measurement job and create a new measurement job.)

Performance measurement management function set in this Recommendation relates to the functions involving PM data collection and reporting specified in 5.3.4 "Performance administration function set" and 5.2.9 "Performance monitoring data accumulation function set" of ITU-T Rec. M.3400 [5].

NMS can set up performance threshold monitors, through which a corresponding QoS alarm may be emitted whenever a performance threshold is crossed. Performance threshold management FS contains the following functions: Create threshold monitor, Delete threshold monitor, Modify threshold monitor, Query threshold monitor, Suspend threshold monitor, Resume threshold monitor and QoS alarm report. Figure 6-8 shows the details.

Performance threshold management function set in this Recommendation relates to the functions involving threshold management specified in 5.3.4 "Performance administration function set" and 5.4.9 "NE(s) performance characterization function set" of ITU-T Rec. M.3400.

Measurement data are network technology dependent and are thus outside the scope of this Recommendation.

6.2.3.2 Performance measurement management function set

6.2.3.2.1 Create measurement job

Summary: NMS can request EMS to create a performance measurement job through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: NMS needs to collect the performance measurement data on some managed entities in order to monitor the network performance, or the performance measurement data can be used for optimizing the network resource in the future.

Description: NMS sends a request to EMS to start a performance measurement job. The parameters may contain: the ID or criteria for the managed entities in which performance data will be collected, the start time of the collection task (optional), the stop time of the collection task (optional), the collection interval of the job, the report interval of the job, the schedule of the job (optional), and the performance parameters to be collected (optional). If the measurement job is started successfully, the job ID will be returned to NMS, and EMS will start the performance collection on the specified network resources according to the parameters of the request. Performance data files are stored in files, and on each reporting interval, the file information will be reported to NMS. Otherwise, it will return error information to NMS.

Exceptions: Invalid parameter; EMS processing error; Communication error.

Post-conditions: A measurement job is started on the request, and it starts to collect and report the corresponding performance data according to their intervals respectively. EMS may send an object creation notification to NMS.

6.2.3.2.2 Delete measurement job

Summary: NMS can request EMS to delete a performance measurement job through the management interface. When a measurement job is deleted, the associated measurement data files are not required to be maintained in EMS.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: NMS does not need the measurement job to collect performance data from the specified managed entities. The history performance data files related to the job in EMS will no longer be used. The specified measurement job exists in EMS and it is stopped or suspended.

Description: NMS sends a request to EMS to delete a performance measurement job. The request parameter is the identifier of the measurement job. If the operation succeeds, the specified measurement job will stop working and the related collecting resources, including the data files, will be released, and EMS will return success information. Otherwise, it will return error information to NMS.

Exceptions: Unknown measurement job; Measurement job not suspended or stopped; EMS processing error; Communication error.

Post-conditions: The specified measurement job is deleted on the request. EMS may send an object deletion notification to NMS.

6.2.3.2.3 Suspend measurement job

Summary: NMS can request EMS to suspend a performance measurement job through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: The specified measurement job exists in EMS and it is not suspended. NMS temporarily does not want the measurement job to collect or report performance data.

Description: NMS can request EMS to suspend a performance measurement job through the management interface. If the operation succeeds, the measurement job will no longer collect and report the corresponding performance data until resumed.

Exceptions: Measurement job already suspended; EMS processing error; Communication error.

Post-conditions: The measurement job is suspended on request, and no performance measurement data associated with this job is collected or reported. EMS may send a state change notification to NMS.

6.2.3.2.4 Resume measurement job

Summary: NMS can request EMS to resume a suspended performance measurement job through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: The specified measurement job exists in EMS and it is suspended. NMS wants the measurement job to collect and report performance data again.

Description: NMS can resume a suspended performance measurement job through the management interface. If the resuming operation succeeds, EMS will return success information and continue to collect and report performance data specified for this job. If the operation fails, it will return error information to NMS.

Exceptions: Measurement job not suspended; EMS processing error; Communication error.

Post-conditions: A measurement job is resumed on request, and it continues to collect and report performance measurement data. EMS may send a state change notification to NMS.

6.2.3.2.5 Query measurement job

Summary: NMS can request EMS to query the parameter values of a performance measurement job through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: NMS needs to query the attribute information of a performance measurement job in EMS. The specified measurement job exists in EMS.

Description: NMS sends a request to EMS to query the parameters of a performance measurement job, which include: the job ID(s), the identifier(s) for the managed objects in which performance data are collected, the start time and stop time of the measurement job, the collection interval of the job, the report interval of the job, the schedule for the job, the administrative state, and the performance parameters to be collected. If the operation succeeds, EMS will return the attribute information of the performance measurement job. If the operation fails, it will return error information to NMS.

Exceptions: EMS processing error; Communication error.

Post-conditions: The corresponding attribute information is returned by EMS as requested.

6.2.3.2.6 Stop measurement job

Summary: NMS can request EMS to permanently stop a performance measurement job through the management interface. When a measurement job is stopped, it does not collect performance data any more, but it still holds the performance data files.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: NMS does not need the measurement job to collect performance data from the specified managed entities. The history performance data files related to the job in EMS may still be used. The specified measurement job exists in EMS.

Description: NMS sends a request to EMS to stop a performance measurement job. If the operation succeeds, the specified measurement job will stop working and EMS will return success information. Otherwise, it will return error information to NMS.

Exceptions: EMS processing error; Communication error.

Post-conditions: The measurement job is stopped on request but the measurement data files are still maintained by this measurement job, which can be retrieved by NMS. EMS may send a state change notification to NMS.

6.2.3.2.7 Performance data report

Summary: Performance data are stored in files. At each report interval, the corresponding performance data file(s) will be prepared by EMS and a "Bulk Data Transfer Ready" notification will be sent to NMS, and then the prepared files will be transferred from EMS to NMS using FTP service. Performance data report function uses "Bulk data transfer function set". See 6.2.1.4.4 for details.

6.2.3.3 Performance threshold management function set

6.2.3.3.1 Create threshold monitor

Summary: NMS can request EMS to create a performance threshold monitor through the management interface.

Assumption: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: NMS needs to monitor some performance measurement parameters collected from managed entities in order to know whether or not there are any performance degradation or service-related performance problems in time. The measurement job(s) containing the measurement parameters to be monitored have been started.

Description: NMS sends a request to EMS to create a performance threshold monitor. The input parameters in the request contain the ID or criteria for the managed entities to be monitored, the monitoring granularity period, and a set of sequence of the name of the measurement parameter, corresponding threshold value, the notifyOnOff switch of alarm notifications, and the related alarm severity (optional). The output parameter is the threshold monitor ID. If the creation succeeds, EMS will return success information. If the operation fails, it will return error information to NMS.

Exceptions: Invalid parameter; EMS processing error; Communication error.

Post-conditions: A performance threshold monitor is created by EMS, and an object creation notification may be reported to NMS. EMS starts to monitor the performance parameters according to the specified threshold values. When a threshold value is crossed, a QoS alarm will be raised.

6.2.3.3.2 Delete threshold monitor

Summary: NMS can request EMS to delete a performance threshold monitor through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: NMS does not need to monitor the specified performance measurement parameters in EMS. The specified threshold monitor exists in EMS and it is suspended.

Description: NMS sends a request to EMS to delete a performance threshold monitor. The parameter is the identifier of the performance threshold monitor. According to the request, EMS will delete the specified performance threshold monitor. If the deletion succeeds, EMS will return success information and no longer monitor the corresponding performance parameters. If the operation fails, it will return error information to NMS.

Exceptions: Unknown Threshold Monitor; Threshold Monitor Not Suspended; EMS Processing Error; Communication Error.

Post-conditions: The specified performance threshold monitor is deleted by EMS, and an object deletion notification may be reported to NMS. EMS will no longer monitor the corresponding performance parameters. If the operation fails, it will return error information to NMS.

6.2.3.3.3 Suspend threshold monitor

Summary: NMS can suspend a performance threshold monitor through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: The specified threshold monitor exists in EMS and it is not suspended. NMS temporarily does not want the threshold monitor to raise any QoS alarms for the performance parameters, or NMS needs to change some attribute values of the monitor.

Description: In this use case, NMS sends a request to suspend a performance threshold monitor. If the suspension succeeds, EMS will return success information and the performance threshold monitor will not act on the corresponding performance parameters and no QoS alarms on the performance parameters will be raised. If the operation fails, it will return error information to NMS.

Exception: Threshold monitor already suspended, EMS processing error, Communication error.

Post-conditions: The performance threshold monitor is suspended according to the request, and it does not monitor the corresponding performance parameters until resumed. A state change notification may be sent to NMS.

6.2.3.3.4 Resume threshold monitor

Summary: NMS can resume a suspended performance threshold monitor through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: The specified measurement job exists in EMS and it is suspended. NMS wants the threshold monitor to act on the performance parameters again.

Description: In this use case, NMS sends a request to resume a suspended performance threshold monitor. If the resumption succeeds, EMS will return success information and the performance

threshold monitor will continue to act on the corresponding performance parameters. If the operation fails, it will return error information to NMS.

Exception: Threshold monitor not suspended, EMS processing error, Communication error.

Post-conditions: The performance threshold monitor is resumed on request and it monitors the performance parameter again. A state change notification may be sent to NMS.

6.2.3.3.5 Modify threshold monitor

Summary: NMS can request EMS to modify the attribute values of a performance threshold monitor through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: The specified threshold monitor exists in EMS and it is suspended. NMS needs to change the attribute values of a threshold monitor in EMS, such as the threshold value and the QoS alarm severities.

Description: NMS sends a request to EMS to modify the attribute values of a performance threshold monitor. Attributes such as the ID or criteria for the managed objects to be monitored, the monitoring granularity period, and the sequence of the name of the measurement parameters, corresponding threshold value, the notifyOnOff switch of alarm notification and the related alarm severity can be modified. According to the request, EMS will modify the performance threshold monitor. If the modification succeeds, EMS will return success information. If the modification fails, it will return error information to NMS.

Exception: Threshold monitor not suspended, Invalid parameter, EMS processing error, Communication error.

Post-conditions: The performance threshold is modified on request. An attribute value change notification may be sent to NMS.

6.2.3.3.6 Query threshold monitor

Summary: NMS can query the information of the performance threshold monitor through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: NMS needs to query the attribute information of a performance threshold monitor from EMS. The specified threshold monitor exists in EMS.

Description: NMS sends a request to EMS to query the attribute information of the threshold monitor in EMS. The information includes: the identifier of the performance threshold monitor, the ID or criteria for the managed objects to be monitored, the monitoring granularity period, and the sequence of the name of the measurement parameter, corresponding threshold value, the notifyOnOff switch of alarm notification and the related alarm severity. If the operation succeeds, EMS will return the requested information. If the operation fails, EMS will send error information to NMS.

Exceptions: EMS processing error; Communication error.

Post-conditions: The requested information of the performance threshold monitor is returned by EMS according to the request.

6.2.3.3.7 QoS alarm report

Summary: When the value of a monitored performance measurement parameter crosses the associated performance threshold value, EMS will trigger a corresponding QoS alarm, which will be reported to NMS if it passes the filtering criteria in the notification dispatcher. This alarm should contain the name of the performance parameter that has been crossed, the corresponding value and the alarm severity. If the alarm severity is specified in the associated performance threshold monitor, the severity of the QoS alarm should refer to the specified value, otherwise the original severity is assigned by EMS. This function uses the "Notification Report Function" in the notification management function set. See 6.2.1.2.1 for details.

6.2.3.4 Query history performance data

Summary: NMS can query history data through the management interface. "Bulk Data Transfer Control Function Set" is used to implement this function. See 6.2.1.4 for details.

6.2.4 Fault management function set

6.2.4.1 Overview

The common parts of the fault management function set include several function sets and functions: ASAP (Alarm Severity Assignment Profile) management FS and alarm report-related functions.

ASAP management function set includes the following functions: create ASAP, delete ASAP, modify ASAP, set ASAP association, remove ASAP association and query ASAP.

ASAP management function set relates to 6.2.1 "Alarm policy function set" of ITU-T Rec. M.3400.

Alarm report-related functions include: configure alarm report filter and alarm report, which relates to 6.2.4 "Alarm reporting function set" of ITU-T Rec. M.3400.

6.2.4.2 ASAP management function set

6.2.4.2.1 Create ASAP

Summary: NMS can request EMS to create an ASAP through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: NMS needs to assign the alarm severities for a set of problems, so that when EMS report alarms, these pre-assigned severities can be referenced in the corresponding alarm notifications.

Description: ASAP is a managed entity used to set the severity of alarms sent by EMS. Through ASAP configuration, NMS can flexibly change alarm severity according to the actual conditions.

NOTE – When ASAP is successfully created, it will not take effect immediately until it is associated with a specified managed object (see "Set ASAP Association" in 6.2.4.2.4). In this use case, NMS sends EMS a request to create an ASAP instance. The request parameter is a list of the problems and their corresponding severity, and a list of the managed entities to be associated with it (can be empty). If the creation operation succeeds, EMS will return the identifier of the ASAP instance as well as success information, and may send an object creation notification to NMS. If the operation fails, EMS will send back error information to NMS.

Exceptions: Invalid parameter; Unknown managed entity; EMS processing error; Communication error.

Post-conditions: An ASAP is successfully created by EMS according to the request. EMS returns the identifier of the ASAP instance and an object creation notification may be sent to NMS. The newly created ASAP will be associated with the managed entities if specified in the request.

6.2.4.2.2 Delete ASAP

Summary: NMS can request EMS to delete an ASAP through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: The specified ASAP exists in EMS, and it is not associated with any managed entities.

Description: In this use case, NMS sends a request to EMS to delete an ASAP. The request parameter is the identifier of the ASAP instance. The ASAP to be deleted should not be associated with any managed object; otherwise it cannot be deleted. If the deletion operation succeeds, EMS returns a success indication, and may send an object deletion notification to NMS. If the operation fails, EMS will send back error information to NMS.

Exceptions: Unknown ASAP; ASAP association not removed; EMS processing error; Communication error.

Post-conditions: The ASAP is successfully deleted by EMS according to the request. EMS may send an object deletion notification to NMS.

6.2.4.2.3 Modify ASAP

Summary: NMS can request EMS to modify, add or delete table entries (problem and the corresponding alarm severity) of an ASAP.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: NMS needs to change the table entries of the alarm severity assignment of an ASAP.

Description: In this use case, NMS sends a request to EMS to modify an ASAP. The request parameter is the new list of the problems and their corresponding severity to be modified.

If the modification operation succeeds, EMS will send success information. If the operation fails, EMS will send back error information to NMS.

Exceptions: Invalid parameter; EMS processing error; Communication error.

Post-conditions: An ASAP is successfully modified by EMS according to the request. EMS may send an attribute value change notification to NMS.

6.2.4.2.4 Set ASAP association

Summary: NMS can request EMS to set or change the association between an ASAP instance and one or more specified managed entities.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: The specified ASAP exists in EMS. NMS needs to set or change the association between an ASAP and one or more managed entities.

Description: When an ASAP is created successfully, it will not take effect immediately until associated with managed entities. When a managed entity is about to report an alarm, it will first check the associated ASAP whether the corresponding alarm severity is specified. If already specified, the corresponding severity is assigned to the alarm and then it is reported to NMS. Otherwise, the original severity is applied. In this use case, NMS sends a request to EMS to set or change the ASAP association. The request parameter is the ID or a list of IDs of the managed

entities to be associated with the ASAP. If the operation succeeds, EMS will return success information, and the ASAP starts to take effect on the specified managed entity(s). If the operation fails, EMS will return error information.

Exceptions: Unknown managed entity; Association already exists; EMS processing error; Communication error.

Post-conditions: The association between the ASAP and the specified managed entity(s) is successfully assigned by EMS. The ASAP then takes effect on the associated managed entity(s).

6.2.4.2.5 Remove ASAP association

Summary: NMS can remove the association between an ASAP and some of its associated managed entity(s).

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: The specified ASAP exists in EMS. The association between the ASAP and the specified managed entity(s) has been assigned. NMS does not want the specified managed entity(s) to refer to this ASAP.

Description: When an ASAP is associated with a managed object, it starts to take effect. When the association between an ASAP and a managed element is no longer needed, it can be removed. If NMS wants to associate a managed object with another ASAP, the association with the previous ASAP must be removed first. In this use case, NMS sends a remove ASAP association request to EMS. The request parameter is: the ID or list of IDs of the managed entity(s) associated with the ASAP. If the operation succeeds, EMS will return success information, and the ASAP associated with specified managed object(s) will not take effect any longer. If the operation fails, EMS will return error information to NMS.

Exceptions: Unknown managed entity; Association not exist; EMS processing error; Communication error.

Post-conditions: According to the NMS request, the association between the ASAP and the specified managed object(s) is removed by EMS. EMS may send the related attribute value change notifications to NMS.

6.2.4.2.6 Query ASAP

Summary: NMS can query the information of an ASAP through the management interface.

Assumptions: The communication between NMS and EMS is available.

Actors: NMS.

Preconditions: NMS needs to query the attribute information of an ASAP in EMS. The specified ASAP exists in EMS.

Description: In this use case, NMS sends a request to EMS to query the attribute information of an ASAP, which includes the ID of the ASAP, the list of the problem and the corresponding severity, and the list of managed entities that have been associated with this ASAP. If the operation succeeds, EMS will return the corresponding attribute values of the ASAP. If the operation fails, EMS will return error information.

Exceptions: EMS processing error; Communication error.

Post-conditions: The corresponding ASAP information is returned by EMS.

6.2.4.3 Alarm report related functions

6.2.4.3.1 Configure alarm report filter

Summary: Configure alarm report filter function uses the notification dispatcher management function. The filtering criteria for alarm type reporting can be specified through modifying a notification dispatcher, which determines whether a specific alarm can be forwarded to NMS or not. For details, please refer to "Notification Dispatcher Management" in 6.2.1.2.2.

6.2.4.3.2 Alarm report

Summary: Alarm report function is described in ITU-T Rec. X.733 [9], which uses the "Notification report" function for alarms. For details, please refer to "Notification report" in Common management function set in 6.2.1.2.1.

6.2.5 Security management function set

For further study.

7 Analysis

This clause provides the detailed analysis of the management interface. In the following subclauses, the related managed entities and their relationships are fully analysed, and the diagrams in these subclauses illustrate the static or dynamic relationships of the managed entities.

7.1 Conventions

In this clause, when specifying managed entities and their management operations, the following abbreviations are applied to indicate the qualifier of attributes, notifications or operation parameters:

– M: Mandatory;
– O: Optional;
– C: Conditional;
– R: Readable;
– W: Writable;
– S: Set by Create.

7.2 Common management function set

7.2.1 Managed entities

7.2.1.1 Class diagram of common management entities

The class diagrams of common management entities are shown below: Figure 7-1 is the inheritance diagram, and Figure 7-2 is the containment diagram.

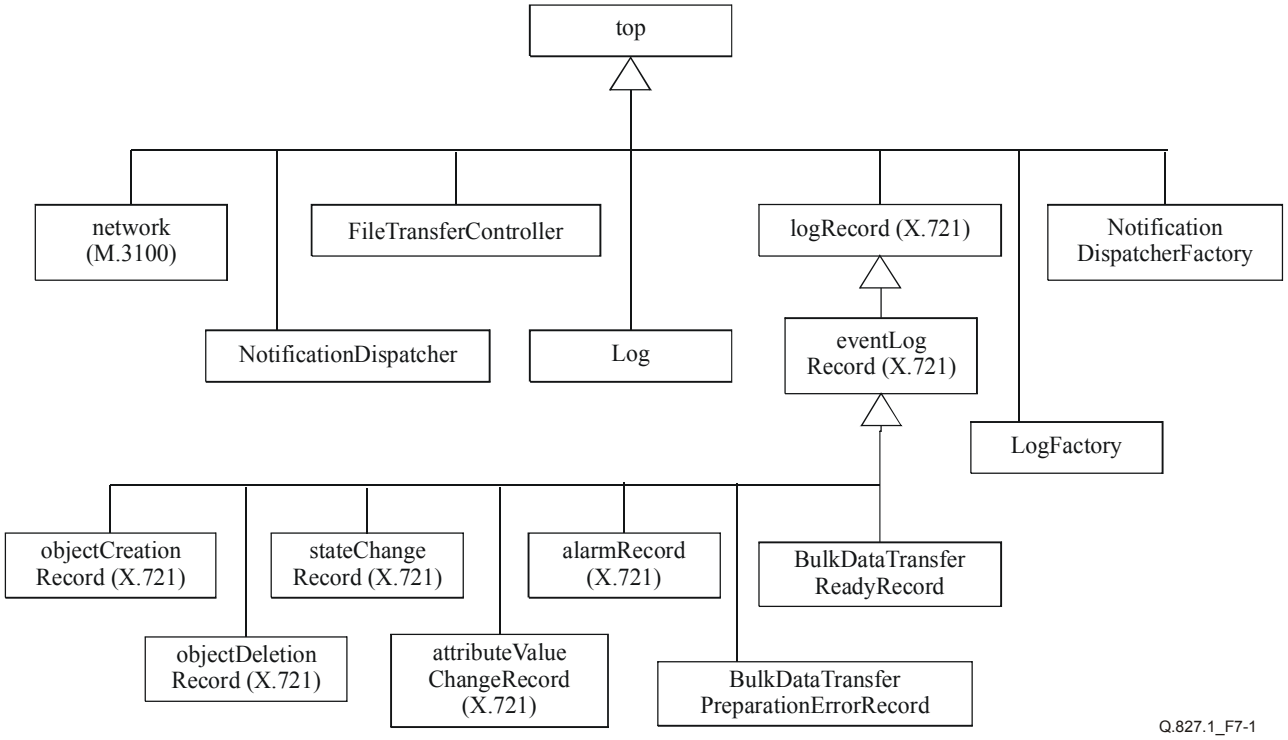

Figure 7-1/Q.827.1 – Inheritance diagram of common management

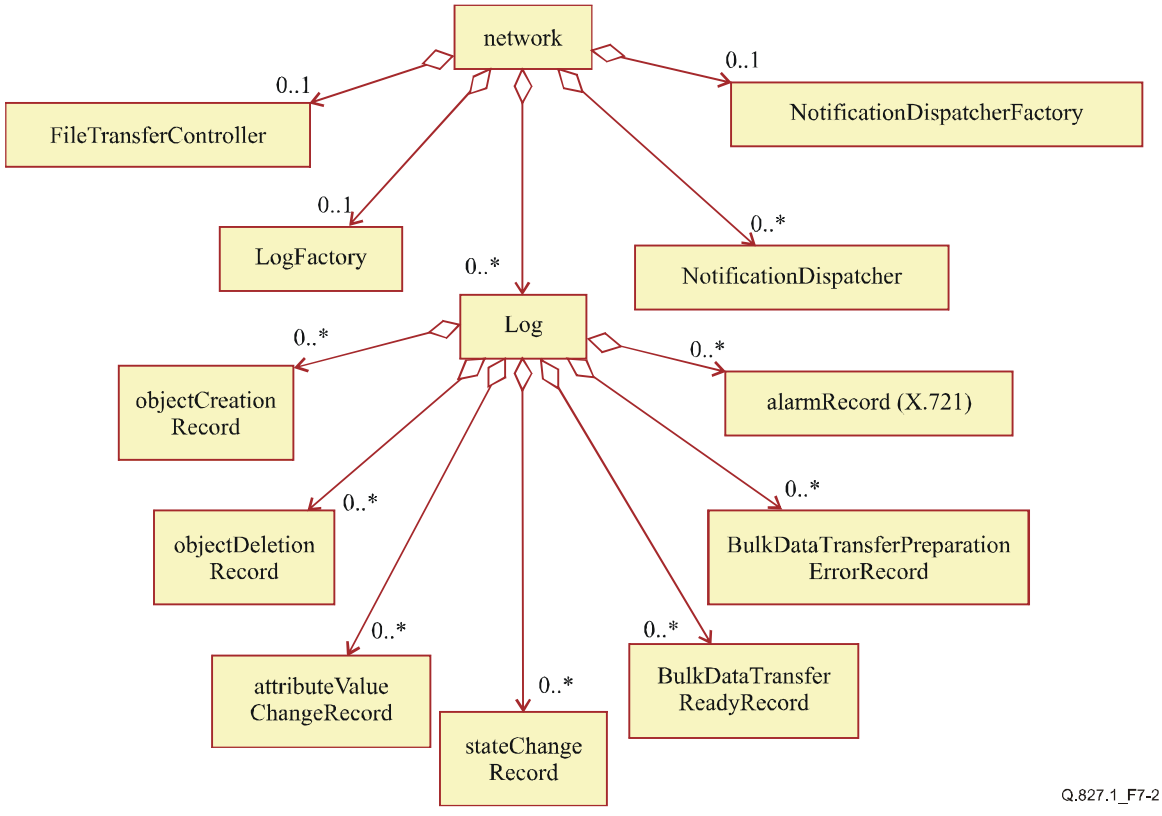

Figure 7-2/Q.827.1 – Containment diagram of common management

7.2.1.2 NotificationDispatcher

Behaviour: This managed entity is used to define the conditions that shall be satisfied by potential event reports before the event report is forwarded to a particular destination(s).			
Attributes			
Name	**Description**	**Type**	**Qualifier**
notificationDispatcherId	This is the unique identifier of the managed entity.	Integer	M, R
discriminatorConstruct	This attribute is composed of one or more assertions and is used to filter the received notifications. If the discriminator construct evaluates to TRUE, the notification dispatcher will send the event to the NMS(s) whose address(es) is defined in "*destinations*" attribute of that notification dispatcher when the dispatcher is permitted to emit notifications. If the value of this attribute is empty, it means the discriminator construct will evaluate to TRUE for any events.	LIST of FilteringCriteria (FilteringCriteria is a list of assertions combined by logical relation operators, each assertion is a string following a predefined expression language.)	M, R/W
administrativeState	This attribute is used to activate (unlock) or deactivate (lock) this managed entity. See ITU-T Rec. X.731 [8] for details. When the value is set to *locked*, the notification dispatcher will no longer emit any event report until it is set to *unlocked again.*	ENUM: {locked, unlocked}	M, R/W
operationalState	This attribute is to indicate the operability of the managed entity, which has two possible values: disabled and enabled. See ITU-T Rec. X.731 for details.	ENUM: {enabled, disabled}	M, R
destinations	This attribute describes the destination(s) to which an event report, if any, will be sent.	LIST of Destination (Destination is the address of the receiving entity.)	M, R
Operations			
Name	**Description**		
suspendNotificationDispatcher	This function is used to suspend a notification dispatcher.		
resumeNotificationDispatcher	This function is used to resume a suspended notification dispatcher.		
modifyNotificationDispatcher	This function is used to modify the attribute values of a notification dispatcher.		
queryNotificationDispatcher	This function is used to query the attribute values of a notification dispatcher.		

Reportable Notifications	
objectCreation	O
objectDeletion	O
attributeValueChange	O
stateChange	O
Relationships: Zero or more instances of this managed entity are contained in one instance of a derived class of network.	

7.2.1.3 NotificationDispatcherFactory

Behaviour: This managed entity is used to create or delete NotificationDispatcher instances.			
Attributes			
Name	**Description**	**Type**	**Qualifier**
factoryId	This is the unique identifier of the managed entity.	Integer	M, R
Operations			
Name	**Description**		
createNotificationDispatcher	This function is used to create a NotificationDispatcher instance.		
deleteNotificationDispatcher	This function is used to delete a NotificationDispatcher instance.		
Relationships: Zero or one instance of this managed entity may be contained in one instance of a derived class of network.			

7.2.1.4 Log

Behaviour: This managed entity is used to define the criteria for controlling the logging of the information in EMS. This managed entity is also used to store incoming event reports and local system notifications.			
Attributes			
Name	**Description**	**Type**	**Qualifier**
logId	This is the unique identifier of the managed entity.	Integer	M, R
discriminatorConstruct	This attribute is composed of one or more assertions and used to filter the received notifications. If the discriminator construct evaluates to TRUE, a Log instance will store the event report it received as a log record if permitted. If this attribute is empty, it means discriminator construct will evaluate to TRUE for any events.	LIST of FilteringCriteria (Filtering Criteria is a list of assertions combined by logical relation operators, each assertion is a string following a predefined expression language.)	M, R/W

Name	Description	Type	Qualifier
administrativeState	This attribute is used to activate (unlock) or deactivate (lock) this managed entity. See ITU-T Rec. X.731 for details. When the value is set to *locked*, this log instance does not store any log records.	ENUM: {locked, unlocked}	M, R/W
operationalState	This attribute is to indicate the operability of the managed entity, which has two possible values: disabled and enabled. See ITU-T Rec. X.731 for details. When this Log instance is unable to store log records for some reasons, the value of this attribute should become *disabled*.	ENUM: {enabled, disabled}	M, R
logFullAction	This attribute specifies the action to be taken when the maximum size of the log has been reached. If the value is set to *wrap*, the oldest records in the log will be deleted to free resources for the creation of new records. If it is set to *halt*, no more records will be logged, and records already in the log will be retained.	ENUM: {wrap, halt}	M, R/W
maxLogSize	This attribute specifies the size of the log measured in number of octets. A log may have an indeterminate size. A max log size of zero shall be used to specify that the log size has no predefined limit.	Integer (Units: octets)	O, R/W
currentLogSize	This attribute specifies the current size of the log measured in octets.	Integer (Units: octets)	O, R
numberOfRecords	This attribute specifies the current number of records contained in the log.	Integer	O, R
capacityAlarmThreshold	This attribute specifies, as a percentage of max log size, the points at which an event will be generated to indicate that a log full or log wrap condition is approaching.	Integer	O, R/W

Operations	
Name	**Description**
suspendLog	This function is used to suspend a log instance.
resumeLog	This function is used to resume a suspended log instance.
modifyLog	This function is used to modify the attribute values of a Log instance.
queryLog	This function is used to query the attribute values of a Log instance.

queryLogRecords	This function is used to query log records in a Log instance.
deleteLogRecords	This function is used to delete log records in a Log instance according to some filtering criteria.

Reportable Notifications	
objectCreation	O
objectDeletion	O
attributeValueChange	O
stateChange	O
processingErrorAlarm	M

Relationships:

Zero or more instances of this managed entity may be contained in one instance of a derived class of network.

7.2.1.5 LogFactory

Behaviour:

This managed entity is used to create or delete Log instances.

Attributes

Name	Description	Type	Qualifier
factoryId	This is the unique identifier of the managed entity.	Integer	M, R

Operations

Name	Description
createLog	This function is used to create a Log instance.
deleteLog	This function is used to delete a Log instance.

Relationships:

Zero or one instance of this managed entity may be contained in one instance of a derived class of network.

7.2.1.6 network (ITU-T Rec. M.3100)

Behaviour:

This managed object class is defined in ITU-T Rec. M.3100 [3]. It represents collections of interconnected telecommunications and management objects (logical or physical) capable of exchanging information. These objects have one or more common characteristics, for example, they may be owned by a single customer or provider, or associated with a specific service network.

7.2.1.7 logRecord (ITU-T Rec. X.721)

Behaviour:

This managed object class is defined in ITU-T Rec. X.721 [7] and used to define the records contained in a log. This managed object class is just used for inheritance purpose in this Recommendation.

7.2.1.8 eventLogRecord (ITU-T Rec. X.721)

Behaviour:

This managed object class is defined in ITU-T Rec. X.721 and used to define the information stored in the log as a result of receiving notifications or event reports. The managed object class inherits from logRecord mentioned above.

This managed object class is just used for inheritance purpose in this Recommendation.

7.2.1.9 alarmRecord (ITU-T Rec. X.721)

Behaviour:
This managed object class is defined in ITU-T Rec. X.721 and used to define the information stored in the log as a result of receiving alarm notifications or alarm reports. This managed object class inherits from eventLogRecord.
Relationships:
Zero or more instances of this managed object may be contained in an instance of Log managed object.

7.2.1.10 attributeValueChangeRecord (ITU-T Rec. X.721)

Behaviour:
This managed object class is defined in ITU-T Rec. X.721 and used to define the information stored in the log as a result of receiving attribute value change notifications or attribute value change event reports. The managed object class inherits from eventLogRecord.
Relationships:
Zero or more instances of this managed entity may exist for an instance of Log managed object.

7.2.1.11 stateChangeRecord (ITU-T Rec. X.721)

Behaviour:
This managed object class is defined in ITU-T Rec. X.721 and used to define the information stored in the log as a result of receiving state change notifications or state change event reports. The managed object class inherits from eventLogRecord.
Relationships:
Zero or more instances of this managed entity may exist for an instance of Log managed object.

7.2.1.12 objectCreationRecord (ITU-T Rec. X.721)

Behaviour:
This managed object class is defined in ITU-T Rec. X.721 and used to define the information stored in the log as a result of receiving object creation notifications or object creation event reports. The managed object class inherits from eventLogRecord.
Relationships:
Zero or more instances of this managed entity may be contained in an instance of Log managed object.

7.2.1.13 objectDeletionRecord (ITU-T Rec. X.721)

Behaviour:
This managed object class is defined in ITU-T Rec. X.721 and used to define the information stored in the log as a result of receiving object deletion notifications or object deletion event reports. The managed object class inherits from eventLogRecord.
Relationships:
Zero or more instances of this managed entity may be contained in an instance of Log managed object.

7.2.1.14 BulkDataTransferReadyRecord

Behaviour:
This managed object class is used to define the information stored in the log as a result of receiving bulk data transfer ready notifications. The managed object class inherits from eventLogRecord.
Attributes:
This managed object inherits all the attributes defined in eventLogRecord, and also contains the attributes as defined in "BulkDataTransferReady" notification. See 7.2.3.2 for details.
Relationships:
Zero or more instances of this managed entity may be contained in an instance of the Log managed object.

7.2.1.15 BulkDataTransferPreparationErrorRecord

Behaviour:
This managed object class is used to define the information stored in the log as a result of receiving bulk data transfer preparation error notifications. The managed object class inherits from eventLogRecord.
Attributes:
This managed object inherits all the attributes defined in eventLogRecord, and also contains the attributes as defined in "BulkDataTransferPreparationError" notification. See 7.2.3.2 for details.
Relationships:
Zero or more instances of this managed entity may be contained in an instance of the Log managed object.

7.2.1.16 FileTransferController

Behaviour:			
This managed entity provides NMS the controlling functions for file transfer between NMS and EMS.			
Attributes			
Name	**Description**	**Type**	**Qualifier**
fileTransferControllerId	This is the unique identifier of the managed entity.	Integer	M, R
Operations			
Name	**Description**		
requestFilePreparation	This operation is used to request the preparation of bulk data files for subsequent transfer via FTP services.		
confirmFileReceived	This operation is used to inform EMS that one or more files have been transferred.		
Relationships:			
Zero or one instance of this managed entity may be contained in an instance of a derived class of network (the root of an EMS).			
Reportable Notifications			
BulkDataTransferReady			M
BulkDataTransferPreparationError			M

7.2.2 Notification dispatcher management function

7.2.2.1 Class diagram

The following Figure 7-3 is the class diagram of Notification Dispatcher.

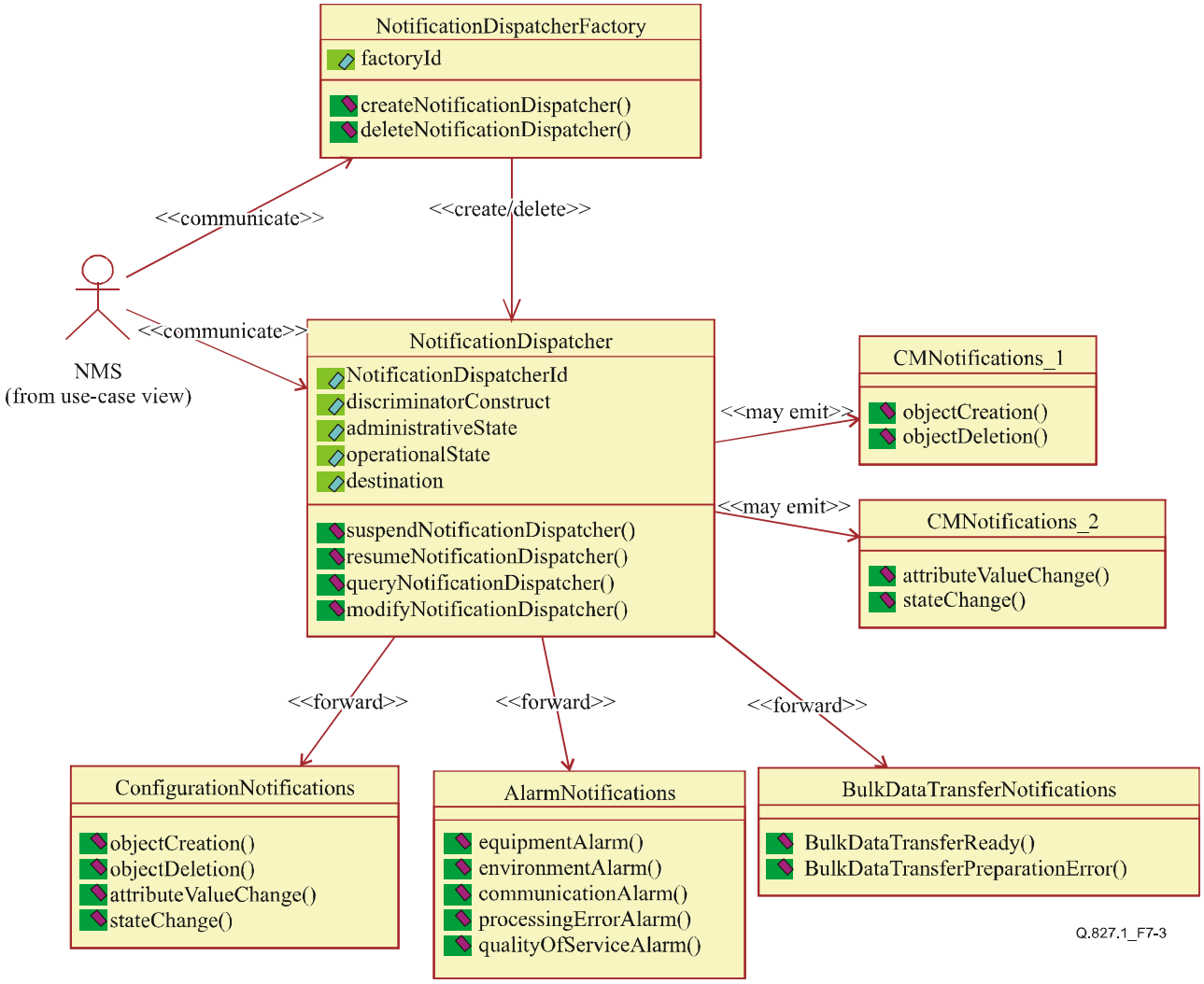

Figure 7-3/Q.827.1 – Class diagram of notification dispatcher

7.2.2.2 Sequence diagram

The following Figure 7-4 is the sequence diagram of Notification Dispatcher management functions.

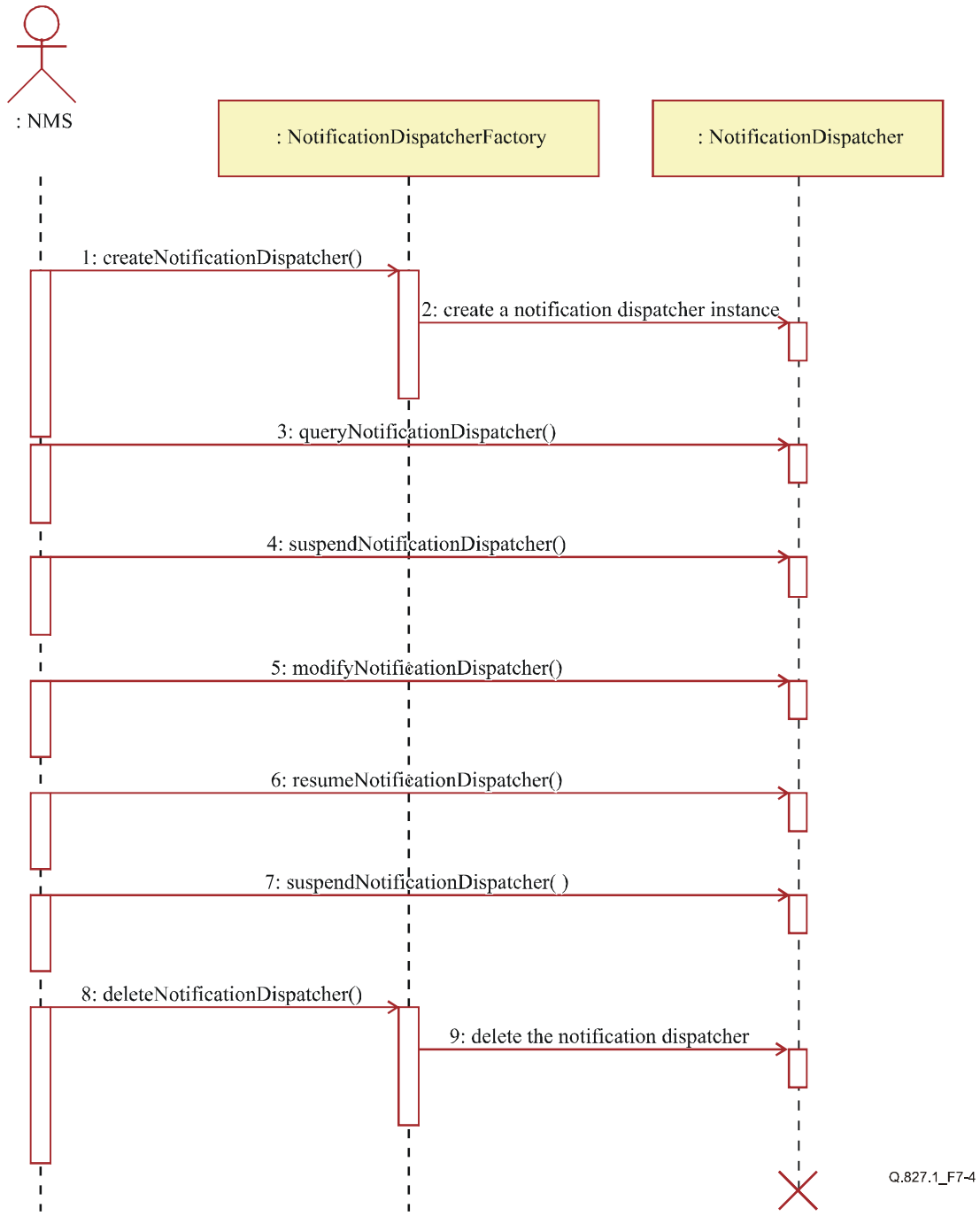

Figure 7-4/Q.827.1 – Sequence diagram of notification dispatcher management functions

7.2.2.3 Management operations

1) "createNotificationDispatcher" Operation

Owner Entity	NotificationDispatcherFactory
Description	This function is used to create a notification dispatcher. The parameters in the request include the destination for event forwarding, the initial administrative state and the discriminator construct of the dispatcher. If the operation succeeds, it will return the identifier of the dispatcher as well as success information. Otherwise, it will return error information.

Operation fields	Name	Description	Type
Input parameters	destinations	This attribute describes the destination(s) to which an event report, if any, will be sent.	LIST of Destination (Destination is the address of the receiving entity.)
	administrativeState	This parameter specifies the initial administrative state of the notification dispatcher.	ENUM: {locked, unlocked}
	discriminatorConstruct	This attribute is composed of one or more assertions and used to filter the received notifications for forwarding. (The type "FilteringCriteria" is a list of assertions combined by logical relation operators, each assertion is a string following a predefined expression language.)	LIST of FilteringCriteria
Output Parameters	notifyDispatcherId	This is the unique identifier of the NotificationDispatcher instance.	Name
Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	At least one input parameter is invalid.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

2) "deleteNotificationDispatcher" Operation

Owner Entity	NotificationDispatcherFactory		
Description	This function is used to delete a notification dispatcher. The parameter in the request is the ID of the notification dispatcher object. If the operation succeeds, it will return success information. Otherwise, it will return error information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	notificationDispatcherId	This parameter specifies the NotificationDispatcher instance.	Name
Output Parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	UnknownDispatcher	The dispatcher specified in the request is unknown to EMS.	
	DispatcherNotSuspended	The specified dispatcher is not suspended before it is deleted.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

3) "suspendNotificationDispatcher" Operation

Owner Entity	NotificationDispatcher		
Description	This function is used to suspend a notification dispatcher object. If the operation succeeds, it will return success information and the notification dispatcher will stop forwarding any notifications. Otherwise, it will return error information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–
Output Parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	DispatcherAlreadySuspended	The notification dispatcher object has already been suspended.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

4) "resumeNotificationFilter" Operation

Owner Entity	NotificationDispatcher		
Description	This function is used to resume a suspended notification dispatcher. If the operation succeeds, it will return success information and the notification dispatcher will begin to forward notifications again. If the operation fails, the operation will return error indication information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–
Output Parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	DispatcherNotSuspended	The dispatcher is not suspended.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

5) "modifyNotificationDispatcher" Operation

Owner Entity	NotificationDispatcher		
Description	This function is used to modify attribute values of a notification dispatcher. If the modification succeeds, it will return success information and the dispatcher will use new criteria and destination(s) to forward notifications. If the modification fails, it will return error indication information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	attributeNamesAndValues	This parameter specifies the attribute names and their values to be set to the dispatcher. The attributes that can be modified include the filtering criteria and the destination.	LIST of Name/Value pairs
Output Parameters	–	–	–
Return Value	–	Success indication	Boolean

Exceptions raised	InvalidParameter	The input parameter is invalid.
	DispatcherNotSuspended	The dispatcher is not suspended.
	EMSProcessingError	Error occurs during EMS processing.
	CommunicationError	Communication error occurs.

6) "queryNotificationDispatcher" Operation

Owner Entity	NotificationDispatcher		
Description	This function is used to query the attribute values of a notification dispatcher. The parameter in the request includes the names of the attributes to be queried. If the function succeeds, it will return the corresponding attribute values. Otherwise, it will return error indication information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	attributeNameList	This parameter specifies the names of the attributes to be queried. The possible attributes can be found in 7.2.1.2.	LIST of Attribute Names
Output Parameters	attributeValues	This parameter specifies the list of attribute names and the corresponding values of the requested attributes.	List of Attribute Name/value pairs
Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	The input attribute name parameter is invalid or unknown to EMS.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

7.2.2.4 Related notifications

1) objectCreation
2) objectDeletion
3) attributeValueChange
4) stateChange

7.2.3 Notification report function

7.2.3.1 Notification header definition

In this clause, the notification header is defined for all the possible notifications in this Recommendation. The content and format of notification header is provided in the following table, and the parameters of the notification header will be used in all the notifications.

Parameters	**Description**	**Type**	**Qualifier**
moInstance	It specifies the managed object instance in which the network event occurred.	Name	M

Parameters	Description	Type	Qualifier
notificationId	This is an identifier for the notification, which may be used to correlate notifications. The identifier of the notification shall be chosen to be unique across all notifications of a particular managed object throughout the time that correlation is significant; it uniquely identifies the notification from other notifications generated by the managed object.	String	O
eventTime	It indicates the event occurrence time.	GeneralizedTime	M
notificationType	It identifies the type of notification. The possible values for this parameter are: { objectCreation, objectDeletion, attributeValueChange, stateChange, equipmentAlarm, environmentalAlarm, communicationAlarm, processingErrorAlarm, qualityOfServiceAlarm, BulkDataTransferReady, BulkDataPreparationError. } The values may be extended when new notification types are used in technology specific network management interfaces.	Integer (Each notification type listed in the left column will be assigned an integer value.)	M

7.2.3.2 Notification definitions

In this clause, all the possible notifications are defined, and the content and format are also provided. For each notification type listed in this clause, the content shall also include the parameters described in 7.2.3.1, "Notification header definition".

1) objectCreation

Parameters	Description	Type	Qualifier
sourceIndicator	It identifies the source of the creation of this managed object instance, which may be caused by resource operation, or by management operations. The value can even be unknown when it cannot be decided.	ENUM {resourceOperation, managementOperation, unknown}	O
attributeList	It contains the list of attribute name and value pairs of the object instance.	LIST of STRUCT {attributeName: String, attributeValue AttributeType} (NOTE – "AttributeType" depends on the attribute name.)	M

2) objectDeletion

Parameters	Description	Type	Qualifier
sourceIndicator	It identifies the source of the deletion of this managed object instance, which may be caused by resource operation, or by management operations. The value can be unknown when it cannot be decided.	ENUM {resourceOperation, managementOperation, unknown}	O

3) attributeValueChange

Parameters	Description	Type	Qualifier
sourceIndicator	It identifies the source of the attribute(s) value change of this managed object instance, which may be caused by resource operation, or by management operations. The value can be unknown when it cannot be decided.	ENUM {resourceOperation, managementOperation, unknown}	O
attributeValue ChangeDefinition	It identifies the list of attributes whose values have changed. It is a set of structure, composed of attributeName, oldAttributeValue (optional) and newAttributeValue.	LIST of STRUCT {attributeName: String; oldAttributeValue: AttributeType, newAttributeValue: AttributeType}	M

4) stateChange

Parameters	Description	Type	Qualifier
sourceIndicator	It identifies the source of the state change of this managed object instance, which may be caused by resource operation, or by management operations. The value can be unknown when it cannot be decided.	ENUM {resourceOperation, managementOperation, unknown}	O
stateChange Definition	It identifies the list of state attributes whose values have changed. It is a set of structure, composed of attributeName, oldAttributeValue (optional) and newAttributeValue.	LIST of STRUCT {attributeName: String; oldAttributeValue: AttributeType, newAttributeValue: AttributeType}	M

5) equipmentAlarm

6) environmentalAlarm

7) communicationAlarm

8) qualityOfServiceAlarm

9) processingErrorAlarm

Parameters	Description	Type	Qualifier
alarmId	It uniquely identifies this alarm from all other alarms generated by EMS.	String (AlarmId)	M
probableCause	It indicates the probable cause of this alarm.	Integer	M
specificProblem	This parameter, when present, identifies further refinements to the probable cause of the alarm.	Integer	O
perceivedSeverity	This attribute indicates the perceived severity of this alarm.	ENUM: { indeterminate, critical, major, minor, warning, cleared}	M
correlatedNotifications	It indicates a set of all notifications to which this notification is considered to be correlated. If the severity is "cleared", it indicates a set of cleared alarms. To a changed alarm, it indicates a set of correlated alarms.	LIST of NotificationId	O
trendIndication	This parameter is used to indicate the trend of this alarm, which can be: less severe, no change, or more severe.	ENUM {lessSevere, noChange, moreSevere}	O
thresholdInfo	This parameter shall be present when the alarm is a result of crossing a threshold. It consists of four sub-parameters: – triggered threshold: The identifier of the threshold attribute that caused the notification; – threshold level: In the case of a gauge, the threshold level specifies a pair of threshold values, the first being the value of the crossed threshold and the second, its corresponding hysteresis; in the case of a counter the threshold level specifies only the threshold value; – observed value: The value of the gauge or counter which crossed the threshold. This may be different from the threshold value if, for example, the gauge may only take on discrete values; – arm time: For a gauge threshold, the time at which the threshold was last re-armed, namely the time after the previous threshold crossing at which the hysteresis value of the threshold was exceeded, thus again permitting generation of notifications when the threshold is crossed. For a counter threshold, the last time at which the threshold offset was applied, or the time at	ThresholdInfoType (STRUCT) See left column for details.	O

Parameters	Description	Type	Qualifier

	which the counter was last initialized (for resettable counters).
	The type of this parameter is as follows:
	ThresholdInfoType ::= STRUCT {
	triggeredThreshold: AttributeIdType;
	observedValue: ObservedValueType,
	thresholdLevel: ThresholdLevelIndType (optional),
	alarmTime: GeneralizedTime (optional)
	}
	ThresholdLevelIndType ::= CHOICE {
	up: STRUCT {
	high: ObservedValueType,
	low: ObservedValueType (optional),
	},
	down: STRUCT {
	high: ObservedValueType,
	low: ObservedValueType,
	}
	ObservedValueType ::= Choice {Integer, Real}		

additionalText	This parameter may be used to specify additional textual information about this alarm.	String	O

10) BulkDataTransferReady

Parameters	Description	Type	Qualifier
transferId	This is the transfer identifier for this file transfer transaction.	Integer	M
jobId	When the files are performance data files, this parameter is used to specify the measurement job.	Integer	O

Parameters	Description	Type	Qualifier

fileInfoList	All information pertaining to the files are provided in this parameter. It is a structure containing fileDirectory and a list of fileName, fileSize, fileCompression, fileCreationTime, and estimated fileDeletionTime.
	LIST of STRUCT {
	fileDirectory: String;
	fileInfoList: LIST of STRUCT {
	fileName: String;
	fileSize: Integer;
	fileCompression: String;
	fileCreationTime: GeneralizedTime;
	fileDeletionTime: GeneralizedTime;
	}
	}
		LIST of STRUCT (see left column)	M

ipAddress	The IP address of the host machine where the files are located.	String	M
userName	The user name to be used in FTP.	String	M
password	The password to be used in FTP.	String	M

11) BulkDataTransferPreparationError

Parameters	Description	Type	Qualifier
transferId	This is the transfer identifier for this file transfer transaction.	Integer	M
jobId	When the files to be prepared are performance data files, this parameter is used to specify the measurement job.	Integer	O
probableCause	This indicates error reason why bulk data transfer preparation failed. The possible probable cause is defined in ITU-T Rec. X.721.	Integer	M
perceivedSeverity	This indicates the severity of the error, can be one of the following: major, minor, warning.	ENUM : {major, minor, warning }	M
additionalText	This parameter may be used to specify additional textual information about this error.	String	O

7.2.4 Log management function

7.2.4.1 Class diagram

The following Figure 7-5 illustrates the class diagram of Log.

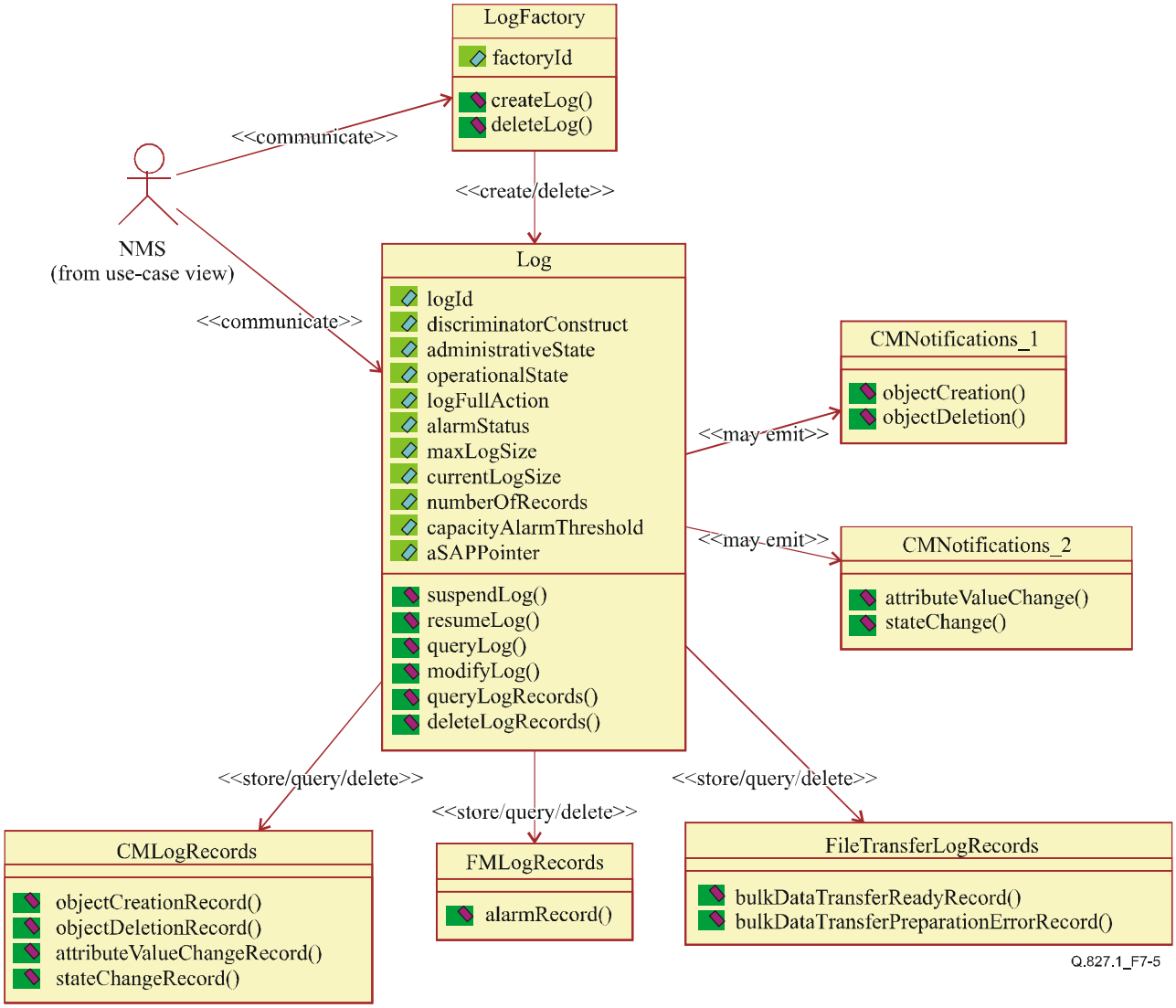

Figure 7-5/Q.827.1 – Class diagram of log

7.2.4.2 Sequence diagram

The following Figure 7-6 illustrates the sequence diagram of log management.

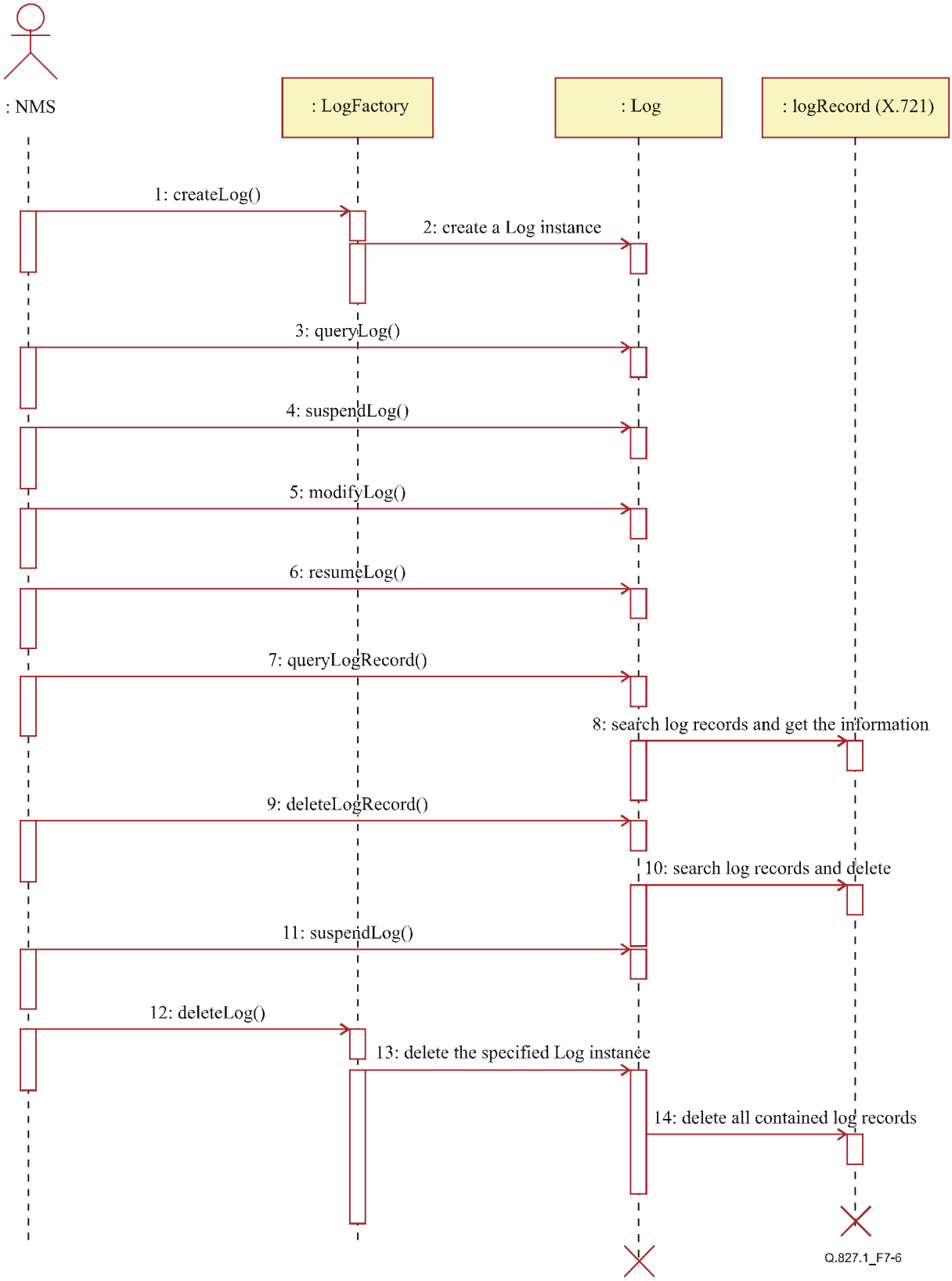

Figure 7-6/Q.827.1 – Sequence diagram of Log management functions

7.2.4.3 Management operations

1) "createLog" operation

Owner Entity	LogFactory		
Description	This function is used to create a log object. The parameters in the request include the initial administrative state, max log size, log full action, capacity threshold, and discriminator construct of the log object. If the operation succeeds, it will return the identifier of the log as well as success information. Otherwise, it will return error information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	administrativeState	This parameter specifies the initial administrative state of the Log object.	ENUM: {locked, unlocked}
	discriminatorConstruct	This attribute is composed of one or more assertions and used to filter the received notifications for logging. (The type "FilteringCriteria" is a list of assertions combined by logical relation operators; each assertion is a string following a predefined expression language.)	LIST of FilteringCriteria
	maxLogSize	This parameter specifies the max size of the log in octets. An empty value indicates there is no limit.	Integer (Units: octets)
	logFullAction	This parameter indicates the action this log will take when it is full, which can be wrap or halt.	ENUM {wrap, halt}
	capacityAlarmThreshold	This parameter specifies the percentage point at which an event will be generated to indicate that a log full or log wrap condition is approaching.	Integer
Output parameters	logId	This parameter specifies the log object.	Name
Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	At least one input parameter is invalid.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

2) "deleteLog" Operation

Owner Entity	LogFactory		
Description	This function is used to delete a log object. The parameter in the request is the ID of the log object. If the operation succeeds, it will return success information. Otherwise, it will return error information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	logId	This parameter specifies the Log object.	Name
Output parameters	–	–	–

Return Value	–	Success indication	Boolean
Exceptions raised	UnknownLog	The Log object specified in the request is unknown to EMS.	
	LogNotSuspended	The Log object is not suspended before it is deleted.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

3) "suspendLog" Operation

Owner Entity	Log		
Description	This function is used to suspend a log object. If the operation succeeds, it will return success information, and the specified log will not record any event until resumed. Otherwise, it will return error information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–
Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	LogAlreadySuspended	The Log object has already been suspended.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

4) "resumeLog" Operation

Owner Entity	Log		
Description	This function is used to resume a suspended log. If the operation succeeds, it will return success information and the log object will begin to record events again. If the operation fails, the function will return error indication information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–
Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	LogNotSuspended	The log object is not suspended.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

5) "modifyLog" Operation

Owner Entity	Log		
Description	This function is used to modify attribute values of a log. If the modification succeeds, it will return success information and the log will use new filtering criteria to record events and behaves according to the new attribute values. If the modification fails, it will return error information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	attributeValues	This parameter specifies the list of attribute names and the corresponding values to be set to the log object. The attributes that can be changed can be found in log object definition in 7.2.1.4.	LIST of Name/Value pairs

Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	The input parameter is invalid.	
	LogNotSuspended	The log object is not suspended.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

6) "queryLog" Operation

Owner Entity	Log		
Description	This function is used to query the attribute values of a log. The parameters in the request include the attribute names to be queried. If the function succeeds, it will return the corresponding attribute values. Otherwise, it will return error information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	attributeNames	This parameter specifies the names of the attributes to be queried.	LIST of AttributeNames
Output parameters	attributeValues	This parameter specifies the names and the corresponding values of the requested attributes.	LIST of Name/value pairs
Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	The input parameter is invalid.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

7) "queryLogRecords" Operation

Owner Entity	Log		
Description	This function is used to query log records. The parameters in the request are the filtering criteria and the time boundary. If the query succeeds, EMS will return the corresponding log records that satisfy the criteria and the time boundary. Otherwise, if the query fails, EMS will return error information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	filteringCriteria	This parameter specifies the filtering criteria of the log records to be queried. (The type "FilteringCriteria" is a list of assertions combined by logical relation operators; each assertion is a string following a predefined expression language.)	LIST of FilteringCriteria
	timeBoundary	This parameter specifies the time boundary during which the records were created.	STRUCT { startTime: Generalized Time; stopTime: Generalized Time; }

Operation fields	Name	Description	Type

Output parameters	logRecords	This parameter specifies the log records satisfying the querying criteria.
		LIST of STRUCT {
		logRecordId: Integer;
		loggingTime: GeneralizedTime;
		recordInfo: RecordInfoType (Choice);
		}
		"RecordInfoType", the type of recordInfo, is a choice of the types of various log records. The possible log records definitions can be found in 7.2.1.
			LIST of STRUCT (see left column)

Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	At least one parameter in the request is invalid.	
	NoSuchLogRecords	There are no log records according to the filtering criteria or time boundary.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

8) "deleteLogRecords" Operation

Owner Entity	Log		
Description	The function is used to delete log records according to some conditions. The parameters in the request are the filtering criteria and the time boundary. If the deletion succeeds, the corresponding records that satisfy the criteria and the time boundary will be deleted. Otherwise, if the deletion fails, EMS will return error indication information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	filteringCriteria	This parameter specifies the filtering criteria of the log records to be deleted. (The type "FilteringCriteria" is a list of assertions combined by logical relation operators; each assertion is a string following a predefined expression language.)	LIST of FilteringCriteria
	timeBoundary	This parameter specifies the time boundary of the records to be deleted.	STRUCT { startTime: Generalized Time; stopTime: Generalized Time; }

Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	At least one parameter in the request is invalid.	
	NoSuchLogRecords	There are no log records according to the criteria or time boundary.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

7.2.4.4 Related notifications

1) objectCreation
2) objectDeletion
3) attributeValueChange
4) stateChange
5) processingErrorAlarm

7.2.5 Bulk data transfer control function set

7.2.5.1 Class diagram

Figure 7-7 is the class diagram of the management entity FileTransferController.

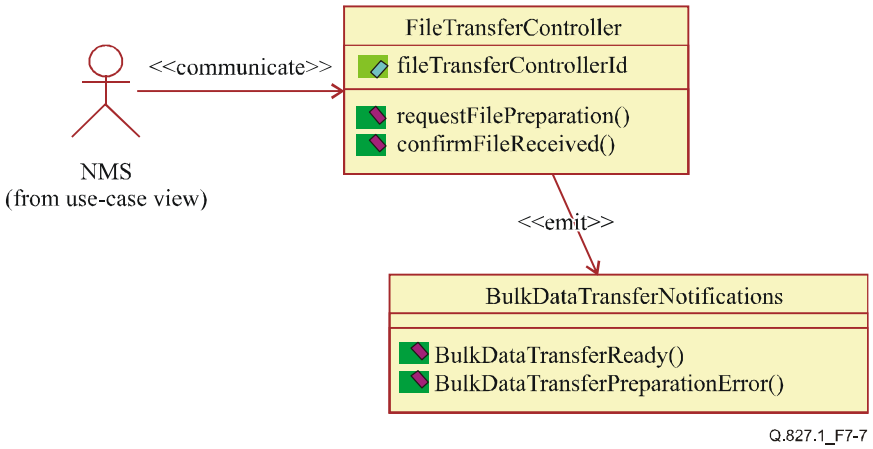

Figure 7-7/Q.827.1 – Class diagram of FileTransferController

7.2.5.2 Sequence diagram

Figure 7-8 depicts the sequence of bulk data transfer controlling functions.

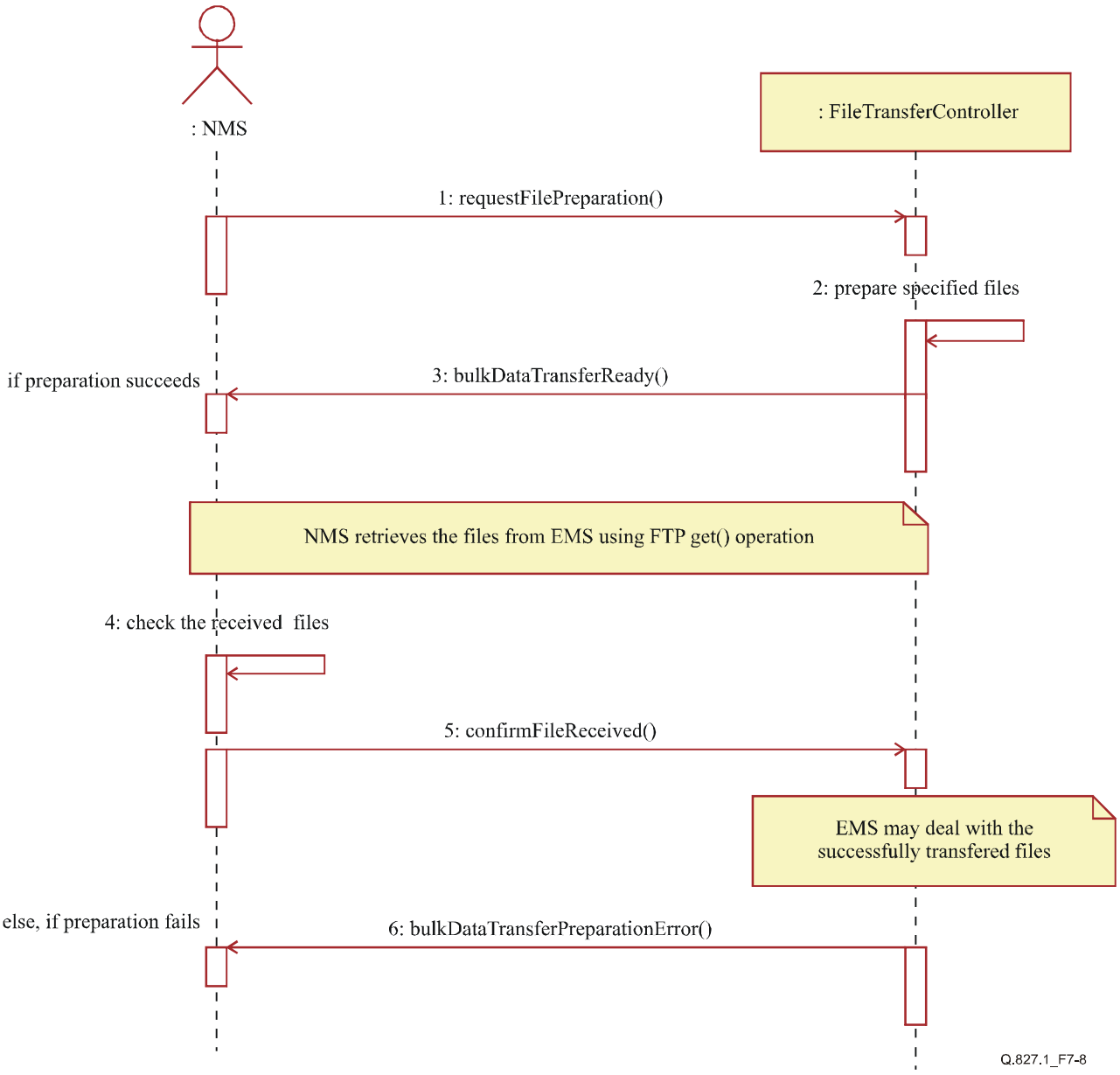

Figure 7-8/Q.827.1 – Sequence diagram of bulk data transfer control

7.2.5.3 Management operations

1) "requestFilePreparation" Operation

Owner Entity	FileTransferController
Description	This operation is used to request the preparation of bulk data files for subsequent transfer via FTP services. In some cases, the data may already exist in the form of one or more files. For other applications, the files must first be produced or formatted for transfer to NMS. If the request from NMS can be accepted, a success response shall be generated by EMS. If, upon receipt of the "requestFilePreparation" operation, EMS is unable to process the request, the appropriate error information shall be returned.

Operation fields	Name	Description	Type
Input parameters	fileType	This parameter specifies the file type of this bulk data transfer, which can be the following cases: 1) Configuration audit file (CM file) 2) Performance data file (PM file) 3) Log record file (Log file)	ENUM {CM, PM, Log}

	objectSelection	This parameter is composed of three parts:
		1) baseMO: indicating the base managed object instance for the object selection;
		2) scope: the scope under the baseMO, which can be "BaseObject", "BaseToNLevel", "IndividualLevel", or "WholeSubTree";
		3) level: this integer indicates the level for the scope, which is only valid when "scope" takes the value of "BaseToNLevel" or "IndividualLevel".
		This parameter is used when the file type attribute is set to "CM" or "Log". This parameter and the following jobId parameter are mutually exclusive.
		When the fileType is "Log", the baseMO should always be the Name of the "Log" instance, and the scope should be "BaseObject", and the level should not be used.
			STRUCT {
			baseMO: Name;
			scope: ENUM;
			level: Integer (optional);
			}

	jobId	This parameter specifies the identifier of a PM measurement job associated with the files to be transferred. This parameter is used only when the file type parameter is "PM". This parameter and the above objectSelection parameter are mutually exclusive. When the fileType is not PM, the value of this parameter should be empty.	Name

Operation fields	Name	Description	Type
Input parameters	timeBoundary	This parameter is only applicable when the "fileType" is "PM" or "Log." This parameter specifies the time boundary for log records or PM files. This parameter is composed of two parts: 1) startTime: the start of the time boundary for log records, or history performance data files. An empty value means there are no constraints; 2) endTime: the end of the time boundary of the information. An empty value means the time when this operation is invoked. When applying this parameter, for log records, the logging time of the records are compared against the time boundary. Only those records whose logging time is within the time boundary will be selected for the file transfer. For PM data, the time stamp when the PM data are generated will be compared against the specified time boundary. When fileType is "CM", the value of this parameter is ignored.	STRUCT { startTime: Generalized Time; endTime: Generalized Time; }
Output parameters	transferId	This parameter identifies the group of operations that together form the control for one bulk data transfer between NMS and EMS. This is the identifier for the file transfer transaction.	Integer
Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	The input parameter is invalid.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

2) "confirmFileReceived" Operation

Owner Entity	FileTransferController		
Description	This operation is used to inform EMS that one or more files have been received through FTP. The files successfully confirmed may then be deleted and the allocated resources may be freed.		
Operation Fields	**Name**	**Description**	**Type**
Input parameters	transferId	This parameter identifies the group of operations that together form the control for one bulk data transfer between NMS and EMS. This is the identifier for the file transfer transaction.	Integer

Operation Fields	Name	Description	Type
	fileReceivedInfo	This parameter indicates the information of file name and file received status for each file listed in "BulkDataTransferReady" notification. This parameter contains a list of structure, which is composed of "filename" and "receivedStatus". The "fileName" is a string specifying the path and name for a file; and the "receivedStatus" is an enumeration which can take the following values: fileOK, fileNotFound, fileFormatInvalid or fileFailedForOtherReason. See 6.2.1.4.3 for the detailed status descriptions.	LIST of STRUCT {fileName: String, receivedStatus: ENUM (see left column for the possible values) }
Output parameters	–	–	–
Return Value	–	void	void
Exceptions raised	UnknownTransferId	The specified transfer ID is unknown to EMS.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

7.2.5.4 Related notifications

1) BulkDataTransferReady
2) BulkDataTransferPreparationError

7.2.5.5 File format definitions

According to the content, file format can be divided into three categories, which are configuration audit file, performance data file, and log record data file.

7.2.5.5.1 Configuration audit file format (CM file)

File Type	CM file		
General Description	The CM file format can be used for configuration auditing purpose. The possible use cases for CM auditing can be found in ITU-T Rec. X.792 [13].		
File Content Fields			
Name	**Description**	**Type**	**Qualifier**
transferId	This information specifies a file transfer transaction. It can be used by NMS to correlate with the operation "requestFilePreparation" it invoked.	Integer	M

Name	Description	Type	Qualifier

moInfoList	This information contains the list of managed object instance information.
	For each MO instance, the information contains the MO class name of the instance, the instance name, and a list of name value pairs of all the configuration attributes of this MO instance.
		LIST of STRUCT {
		moInstance: STRUCT {
		moClasse: String;
		moName: Name;
		}
		attrInfoList: LIST of STRUCT {
		attrName: String;
		attrValue: AttributeType;
		};
		(AttributeType can be applicable for variety of data types, and it depends on the value of attrName.)
			M

7.2.5.5.2 Performance measurement data file format (PM file)

File Type	PM file		
General Description	The PM file format is used for performance measurement data file format.		
File Content Fields			
Name	**Description**	**Type**	**Qualifier**
transferId	This information specifies a file transfer transaction. It can be used by NMS to correlate with a "requestFilePreparation" operation it invoked, or the same parameter provided in the "BulkDataTransferReady" notification.	Integer	M
jobId	This is the identification of the specific measurement job. It may be used by NMS to correlate with a measurement job.	Name	M
granularity Period	This field specifies the time interval of the successive measurement period.	Integer (Units is a choice of minutes, hours, or days)	M
reportingPeriod	This field specifies the time interval of the reporting period for PM data files, which shall be one or multiple times of the granularity period.	Integer (Units is a choice of minutes, hours, or days)	M

Name	Description	Type	Qualifier

measurement IntervalsInfo List	This field specifies a list of performance measurement data for one or more granularity period. For each granularity period the information contains the time stamp of the end of the granularity period, and the measurementInfoList, which can involve more than one MOs on which the measurement is collected. For each MO instance, the information contains the MO class name of the instance, the instance name, and a list of name value pairs of all the performance measurement parameters of this MO instance.	LIST of STRUCT {
		timeStamp: GeneralizedTime;
		measurementInfoList: LIST of STRUCT {
		moInstance: STRUCT {
		moClass: String;
		moName: Name;
		}
		pmAttrInfoList: LIST of STRUCT {
		pmAttrName: String;
		pmAttrValue: Choice of {Integer, Real};
		}
		}
		}
			M

7.2.5.5.3 Log record file format (Log file)

File Type	Log file		
General description	The Log file format defined in this clause is used for log record file format.		
File Content Fields			
Name	**Description**	**Type**	**Qualifier**
transferId	This information specifies a file transfer transaction. It can be used by NMS to correlate this information with the "requestFilePreparation" operation it invoked.	Integer	M
logInstance	This field identifies the log instance.	Name	M

logRecord InfoList	This field provides the requested log records information. The information is a list of log records, and for each log record, the information contains the logRecordId, the logging time, and the stored notification information. The type of each stored notification depends on the notification type.	LIST of STRUCT {
		logRecordId: Integer;
		loggingTime: GeneralizedTime;
		recordInfo: RecordInfoType (Choice);
		}
		"RecordInfoType", the type of recordInfo, is a choice of the types of various log records. The possible log records definitions can be found in 7.2.1.
			M

7.3 Performance management function set

7.3.1 Managed entities

7.3.1.1 Class diagram of performance management entities

The class diagrams of performance management entities are show below: Figure 7-9 is the inheritance diagram, and Figure 7-10 is the containment diagram.

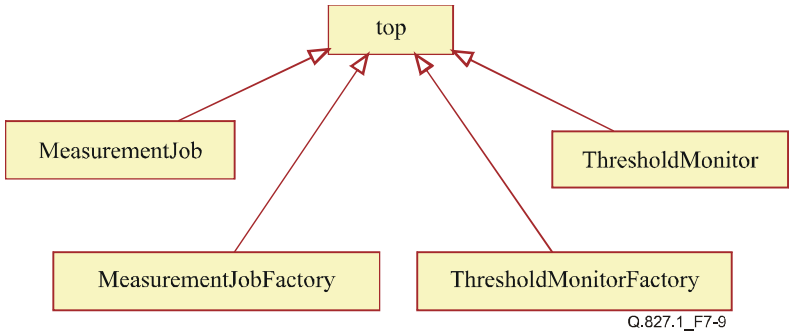

Figure 7-9/Q.827.1 – Inheritance diagram of performance management

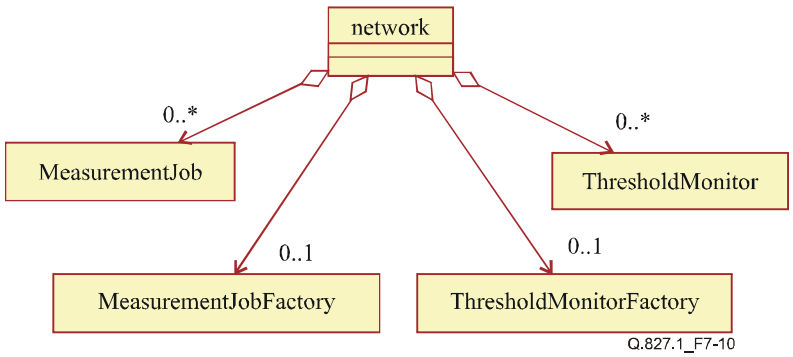

Figure 7-10/Q.827.1 – Containment diagram of performance management

As shown below, Figure 7-11 is the class diagram of the managed entity MeasurementJob, and Figure 7-12 is the class diagram of the managed entity ThresholdMonitor.

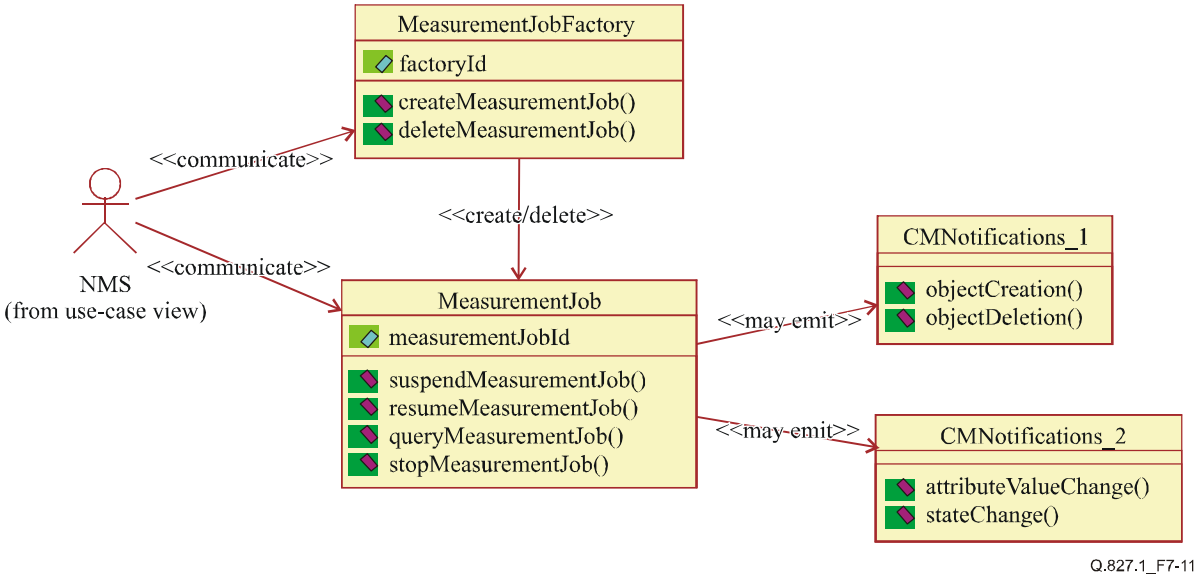

Figure 7-11/Q.827.1 – Class diagram of measurementjob

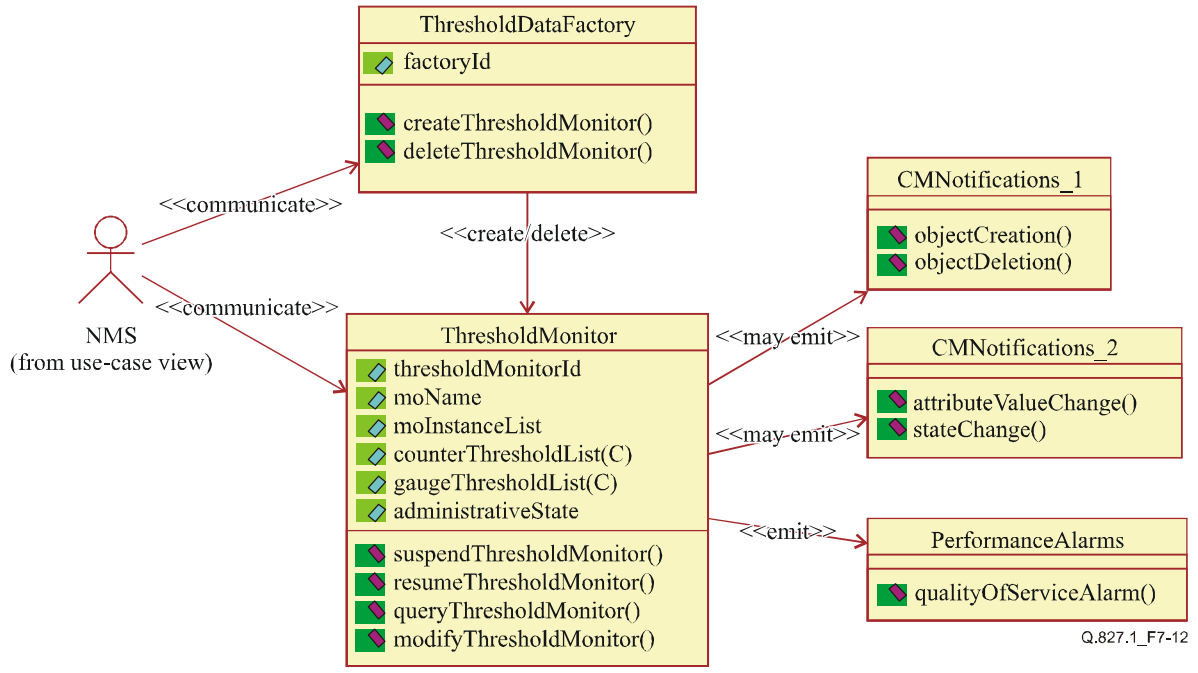

Figure 7-12/Q.827.1 – Class diagram of ThresholdMonitor

7.3.1.2 MeasurementJobFactory

Behaviour: This managed entity is used to create or delete performance measurement job instances.			
Attributes			
Name	**Description**	**Type**	**Qualifier**
factoryId	This is the unique identifier of the factory.	Integer	M, R
Operations			
Name	**Description**		
createMeasurementJob	This function is used to create a performance measurement job instance.		

deleteMeasurementJob	This function is used to delete a performance measurement job instance.
Relationships: Zero or one instance of this managed entity exists in an instance of derived class of network.	

7.3.1.3 MeasurementJob

Behaviour:

This managed entity represents a measurement job which is used to control the collection of the measurement parameters for one or more managed object instances.

Attributes

Name	Description	Type	Qualifier
measurementJobId	This is the unique identifier of the managed entity.	Integer	M, R
moClass	This attribute specifies the class name of managed object instances on which the performance data are collected.	String	M, R/S
moInstanceList	This parameter specifies the MO instances on which the performance data are collected. When this attribute is empty, it indicates all the instances of the same MO class specified in "moClass" parameter.	LIST of Name	M, R/S
granularityPeriod	This attribute specifies the time interval of the measurement period for collection performance data.	Integer (Units is a choice of minutes, hours, or days)	M, R/S
reportingPeriod	This attribute specifies the time interval of the reporting period for performance data, which shall be one or multiple times of the granularity period.	Integer (Units is a choice of minutes, hours, or days)	M, R/S
startTime	This attribute specifies the start time when this performance measurement job takes effect. When empty, it indicates the task will start immediately.	GeneralizedTime	M, R/S
stopTime	This attribute specifies the end time of this performance measurement job. When empty, it indicates there is no automatic stop time for this task, unless it is stopped by another operation.	GeneralizedTime	M, R/S
performanceParameterList	This attribute specifies the performance parameters to be measured.	LIST of AttributeName	M, R/S
schedule	This attribute indicates the detailed schedule for this collection task on daily or weekly basis. See ITU-T Rec. X.721 for details.	Choice of Daily Schedule or Weekly Schedule See ITU-T Rec. X.721.	M, R/S
administrativeState	This attribute is used to activate (unlock) or deactivate (lock) a measurement job.	ENUM {locked, unlocked}	M, R/S

jobStatus	This attribute is used to indicate the running status of a measurement job, which may have the following values: – scheduled: the job is created, but the start time is not reached yet. – active: the job is running actively. – off-duty: the job is off-duty according to the schedule. – suspended: this job is suspended. – stopped: the job has been stopped by NMS or the stop time has been reached. (When more than one of the values are acceptable for the job status, the latter one in the order of the above list has higher priority.)	ENUM {scheduled, active, off-duty, suspended, stopped.}	M, R

Operations	
Name	**Description**
suspendMeasurementJob	This function is used to suspend a performance measurement job.
resumeMeasurementJob	This function is used to resume a suspended performance measurement job.
queryMeasurementJob	This function is used to query the attribute values of a performance measurement job.
stopMeasurementJob	This function is used to stop a performance measurement job.

Relationships:
Zero or more instances of this managed entity exist in an instance of a derived class of network.

Reportable Notifications	
objectCreation	O
objectDeletion	O
stateChange	O

7.3.1.4 ThresholdMonitorFactory

Behaviour:
This managed entity is used to create or delete instances of ThresholdMonitor.

Attributes			
Name	**Description**	**Type**	**Qualifier**
factoryId	This is the unique identifier of the managed entity.	Integer	M, R

Operations	
Name	Description
createThresholdMonitor	This function is used to create a performance threshold monitor instance.
deleteThresholdMonitor	This function is used to delete a performance threshold monitor instance.

Relationships:
Zero or one instance of this managed entity exists in an instance of a derived class of network.

7.3.1.5 ThresholdMonitor

Behaviour:			
This managed entity is used for NMS to set thresholds on some performance measurement parameters. When the value of a specified measurement parameter crosses some threshold value, a qualityOfServiceAlarm will be emitted by this ThresholdMonitor. This entity also provides the controlling functions of the threshold information.			
Attributes			
Name	**Description**	**Type**	**Qualifier**
thresholdMonitorId	This is the unique identifier of the managed entity.	Integer	M, R
moClass	This attribute specifies the class name of managed object instances on which the performance data are collected.	String	M, R/S
moInstanceList	This attribute specifies the MO instances on which the performance data are collected. An empty list means it is applicable for any instance of the managed object class specified in "moClass" attribute.	LIST of Name	M, R/W
Name	**Description**	**Type**	**Qualifier**

counterThresholdList	This attribute contains a set of threshold settings for performance attributes of the counter type (e.g., errored seconds). Each threshold setting consists of the attribute identifier, the threshold value and (optionally) the severity of the threshold-exceeded event.
	The type can be described by the following text:
	CounterThresholdListType ::= List of STRUCT {
	attributeId: string,
	severityIndicatingThreshold: SeverityIndicatingThresholdType
	}
	SeverityIndicatingThresholdType ::= STRUCT {
	threshold: Choice of (Integer, Real);
	notifyOnOff: Boolean;
	severityIndication: Severity (ENUM, optional)
	}
	The Severity is an enumeration which may have one of the following values: {indeterminate, critical, major, minor, warning, cleared}
	The description of thresholdValue, notifyOnOff and severityIndication can be found in ITU-T Rec. X.739 [12].
		Counter Threshold ListType (LIST of STRUCT),
		See the left column for details.
			C (see Note), R/W

gaugeThresholdList	This attribute contains a set of threshold settings for performance attributes of the gauge type. Each threshold setting consists of the attribute identifier, the threshold values (it may have mutiple levels) and (optionally) the severity of the threshold-exceeded event for each level.
	GaugeThresholdAttributeListType::= LIST of STRUCT {
	attributeId AttributeId,
	severityIndicatingGaugeThreshold: LIST of STRUCT {
	notifyLow: SeverityIndicatingThresholdType,
	notifyHigh: SeverityIndicatingThresholdType,
	}
	}
	The Type "SeverityIndicatingThresholdType" can be found in the above row, and the concepts of "notifyLow" and "notifyHigh" can be found in ITU-T Rec. X.739.
		Gauge Threshold Attribute ListType (LIST of STRUCT)
		See the left column for details
			C (see Note), R/W

administrativeState	This attribute is used to activate (lock) or deactivate (unlock) this ThresholdMonitor. When the threshold monitor is locked, no qualityOfServiceAlarm will be emitted from it.	ENUM { locked, unlocked}	M, R/W
monitorGranularityPeriod	This attribute specifies the time period for this ThresholdMonitor instance to check whether the specified performance parameters have crossed the corresponding threshold values.	Integer (Units is a choice of minutes, hours, or days)	M, R/W
Operations			
Name	**Description**		
suspendThresholdMonitor	This function is used to suspend a threshold monitor.		
resumeThresholdMonitor	This function is used to resume a suspended threshold monitor.		
modifyThresholdMonitor	This function is used to modify attribute values of a performance threshold monitor.		
queryThresholdMonitor	This function is used to query the attribute values of a performance threshold monitor.		
Relationships: Zero or more instances of this managed entity exist in an instance of derived class of network.			

Reportable Notifications	
objectCreation	O
objectDeletion	O
stateChange	O
attributeValueChange	O
qualityOfServiceAlarm	M
NOTE – The existence of the attributes "counterThresholdList" and "gaugeThresholdList" is mutually exclusive.	

7.3.2 Sequence diagram

Figure 7-13 is the sequence diagram of performance measurement management, and Figure 7-14 is the sequence diagram of performance threshold management.

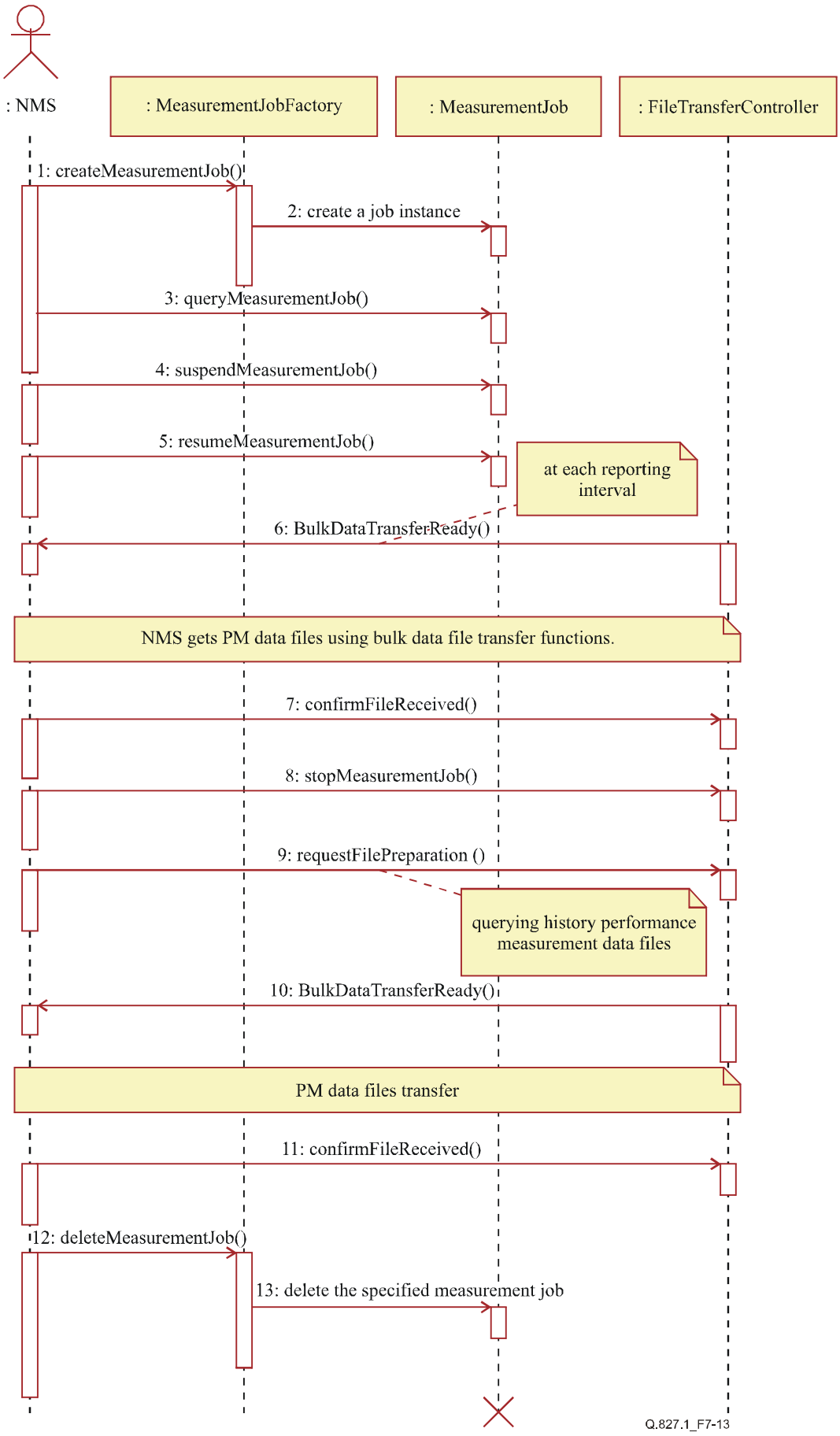

Figure 7-13/Q.827.1 – Sequence diagram of performance measurement management

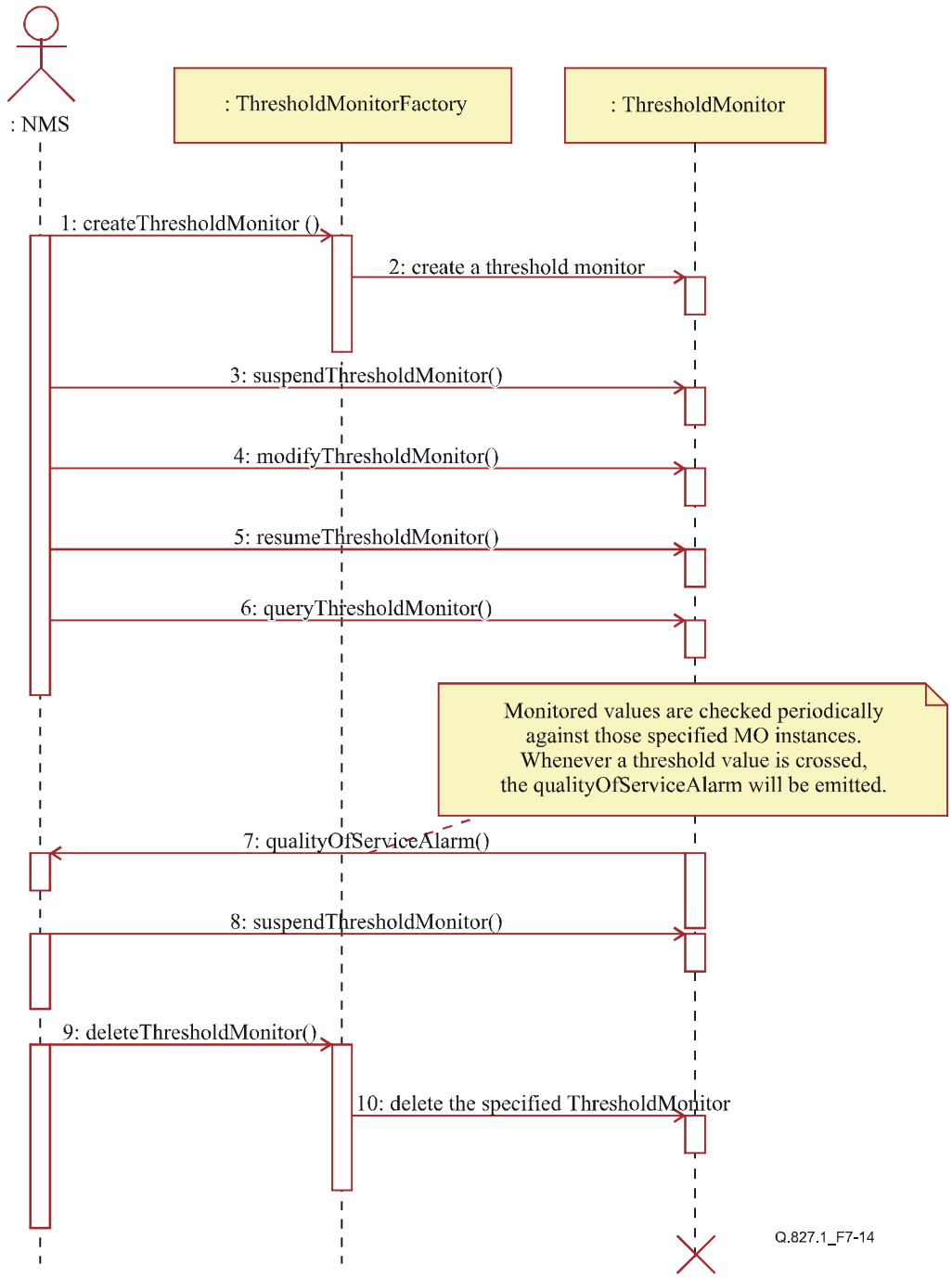

Figure 7-14/Q.827.1 – Sequence diagram of performance threshold management

7.3.3 Management operations

7.3.3.1 Measurement job related management operations

1) "createMeasurementJob" Operation

Owner Entity	MeasurementJobFactory
Description	This function is used to create an instance of performance measurement job. If the creation succeeds, EMS will start to collect and report the performance data according to the request. A measurement job may contain multiple measurement parameters of multiple managed object instances.

Operation fields	Name	Description	Type
Input parameters	moClass	This parameter specifies the class name of managed object instances on which the performance data are collected.	String
	moInstanceList	This parameter specifies the MO instances on which the performance data are collected. When this parameter is empty, it indicates all the instances of the same MO class specified in "moClass" parameter.	LIST of Name
	granularityPeriod	This parameter specifies the time interval of a measurement period for collection performance data.	Integer (Units is a choice of minutes, hours, or days)
	reportingPeriod	This parameter specifies the time interval of the reporting period for performance data, which shall be one or multiple times of the granularity period.	Integer (Units is a choice of minutes, hours, or days)
	startTime	This parameter specifies the start time when this performance measurement job takes effect. When empty, it indicates the task will start immediately.	Generalized Time
	stopTime	This parameter specifies the end time of this performance measurement job. When empty, it indicates there is no automatic stop time for this task, unless it is stopped by another operation.	Generalized Time
	performanceParameterList	This parameter specifies the performance parameters to be measured.	LIST of AttributeName
	schedule	This indicates the detailed schedule for this collection task on daily or weekly basis. See ITU-T Rec. X.721 for details.	Choice of DailySchedule or Weekly Schedule. See ITU-T Rec. X.721.
	administrativeState	This parameter specifies the initial administrative state of the measurement job.	ENUM {locked, unlocked}
Output parameters	measurementJobId	This parameter specifies the identifier of the performance measurement job.	Name

Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	At least one of the input parameters is invalid.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

2) "deleteMeasurementJob" Operation

Owner Entity	MeasurementJobFactory		
Description	This function is used to delete an instance of performance measurement job. If the operation succeeds, EMS will no longer collect the performance data for the specified managed objects. When a measurement job is deleted, all the history data files related to this measurement job are not required to be maintained in EMS.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	measurementJobId	This parameter specifies the identifier of the performance measurement job.	Name
Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	UnknownMeasurementJob	The identifier of the measurement job specified in the request is unknown to EMS.	
	MeasurementJobNotSuspendedOrStopped	The measurement job is not suspended or stopped before it is deleted.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

3) "suspendMeasurementJob" Operation

Owner Entity	MeasurementJob		
Description	This function is used to suspend a performance measurement job.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–
Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	MeasurementJobAlreadySuspended	The measurement job has already been suspended.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

4) "resumeMeasurementJob" Operation

Owner Entity	MeasurementJob		
Description	This function is used to resume a suspended performance measurement job.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–
Output parameters	–	–	–
Return Value	–	Success indication	Boolean

Exceptions raised	MeasurementJobNot Suspended	The measurement job is not suspended.
	EMSProcessingError	Error occurs during EMS processing.
	CommunicationError	Communication error occurs.

5) "queryMeasurementJob" Operation

Owner Entity	MeasurementJob		
Description	This function is used to query the parameter values of a performance measurement job.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–
Output parameters	jobInfo	This parameter specifies the information of the specified measurement job, which will be described by the type name JobInfoType: JobInfoType ::= STRUCT { jobId: Interger; moClass: String; moInstanceList: NameList; granularityPeriod: PeriodType; reportingPeriod: PeriodType; startTime: GeneralizedTime; stopTime: GeneralizedTime; parameterList: StringList; schedule: ScheduleType; jobStatus: ENUM; administrativeState: ENUM; } Where the PeriodType is an integer with the units being a choice of minutes, hours, or days, the value of jobStatus may be one of the following: scheduled, active, off-duty, suspended, stopped.	JobInfoType (see left column for details)
Return Value	–	Success indication	Boolean
Exceptions raised	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

6) "stopMeasurementJob" Operation

Owner Entity	MeasurementJob		
Description	This function is used to stop a performance measurement job. If the operation succeeds, EMS will no longer collect the performance data for the specified managed objects. When a measurement job is stopped, all the history data files related to this measurement job shall still be maintained in EMS.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–
Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

7.3.3.2 Threshold monitor related management operations

1) "createThresholdMonitor" Operation

Owner Entity	ThresholdMonitorFactory		
Description	This function is used to create an instance of performance threshold monitor through the management interface.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	moClass	This parameter specifies the class name of managed object instances on which the performance data are collected.	String
	moInstanceList	This parameter specifies the MO instances on which the performance data are collected. An empty list means it is applicable for any instance of the managed object class specified in "moClass" parameter.	Name List
	thresholdInfoList	This parameter is a list of the thresholds for measurement parameters. It can be either a list of counter-typed thresholds, or a list of gauge-typed thresholds. The detailed description and type definition can be found in the description of the managed entity "ThresholdMonitor" in 7.3.1.5.	Choice {CounterThresholdListType, GaugeThresholdAttributeListType }
	administrativeState	This parameter specifies the initial value of administrative state.	ENUM {locked, unlocked }
	monitorGranularityPeriod	This parameter specifies the period for checking threshold-crossing events.	Integer (Units is a choice of minutes, hours, or days)

Output parameters	thresholdMonitorId	This parameter specifies the identifier of the threshold monitor.	Name
Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	At least one of the input parameters is invalid.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

2) "deleteThresholdMonitor" Operation

Owner Entity	ThresholdMonitorFactory		
Description	This function is used to delete an instance of threshold monitor.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	thresholdMonitorId	This parameter specifies identifier of the threshold monitor to be deleted.	Name
Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	UnknownThresholdMonitor	The threshold monitor ID specified in the request is unknown to EMS.	
	ThresholdMonitorNot Suspended	The threshold monitor is not suspended before it is deleted.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

3) "suspendThresholdMonitor" Operation

Owner Entity	ThresholdMonitor		
Description	This function is used to suspend a threshold monitor.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–
Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	ThresholdMonitorAlready Suspended	The threshold monitor specified in the request has already been suspended.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

4) "resumeThresholdMonitor" Operation

Owner Entity	ThresholdMonitor		
Description	This function is used to resume a suspended threshold monitor.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–
Output parameters	–	–	–
Return Value	–	Success indication	Boolean

Exceptions raised	ThresholdMonitorNot Suspended	The threshold monitor specified in the request is not suspended.
	EMSProcessingError	Error occurs during EMS processing.
	CommunicationError	Communication error occurs.

5) "queryThresholdMonitor" Operation

Owner Entity	ThresholdMonitor		
Description	This function is used to query parameter values of a threshold monitor.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–

Output parameters	thresholdMonitorInfo	This parameter specifies the information of the specified threshold monitor, which includes the following information:
		– moClass: the class name of MO instances on which the performance data are collected.
		– moInstanceList: the MO instances on which the performance data are collected. An empty list means it is applicable for any instance of the managed object class specified in "moClass" parameter.
		– thresholdInfoList: The list of the threshold for measurement parameters. It can be either a list of counter-typed thresholds, or a list of gauge-typed thresholds.
		– administrativeState: The administrative state of the threshold monitor.
		– monitorGratularityPeriod: The period for checking threshold-crossing events.
			STRUCT {
			moClass: String;
			moInstance List: LIST of Name;
			threshold InfoList: Choice {Counter ThresholdList Type, Gauge ThresholdList Type
			};
			administrative State: ENUM;
			monitor Granularity Period: Integer;
			}
			NOTE – The two types for thresholdInfo List can be found in "Threshold Monitor" in 7.3.1.5.

Return Value	–	Success indication	Boolean
Exceptions raised	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

6) "modifyThresholdMonitor" Operation

Owner Entity	ThresholdMonitor		
Description	This function is used to modify the parameter values of a threshold monitor.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	moInstanceList	This parameter specifies the MO instances on which the performance data are collected. An empty list means it is applicable for any instance of the managed object class specified in "moClass" attribute.	LIST of Name
	thresholdInfoList	This parameter is a list of the thresholds for measurement parameters. It can be either a list of counter-typed thresholds, or a list of gauge-typed thresholds. The description and types for this parameter can be found in "ThresholdMonitor" in 7.3.1.5.	Choice {CounterThresholdListType, GaugeThresholdAttributeListType }
	monitorGranularityPeriod	This parameter specifies the period for checking threshold-crossing events.	Integer (Units is a choice of minutes, hours, or days)
Output Parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	ThresholdMonitorNotSuspended	The threshold monitor specified in the request is not suspended before it is modified.	
	InvalidParameter	At least one of the input parameters is invalid.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

7.3.4 Related notifications

1) objectCreation
2) objectDeletion
3) attributeValueChange
4) stateChange
5) qualityOfServiceAlarm
6) BulkDataTransferReady
7) BulkDataTransferPreparationError

7.4 Fault management function set

7.4.1 Managed entities

7.4.1.1 Class diagram of fault management entities

The class diagrams of fault management entities are shown below: Figure 7-15 is the inheritance diagram, Figure 7-16 is the entities relationship diagram, and Figure 7-17 is the class diagram of the management entities ASAP and ASAPFactory.

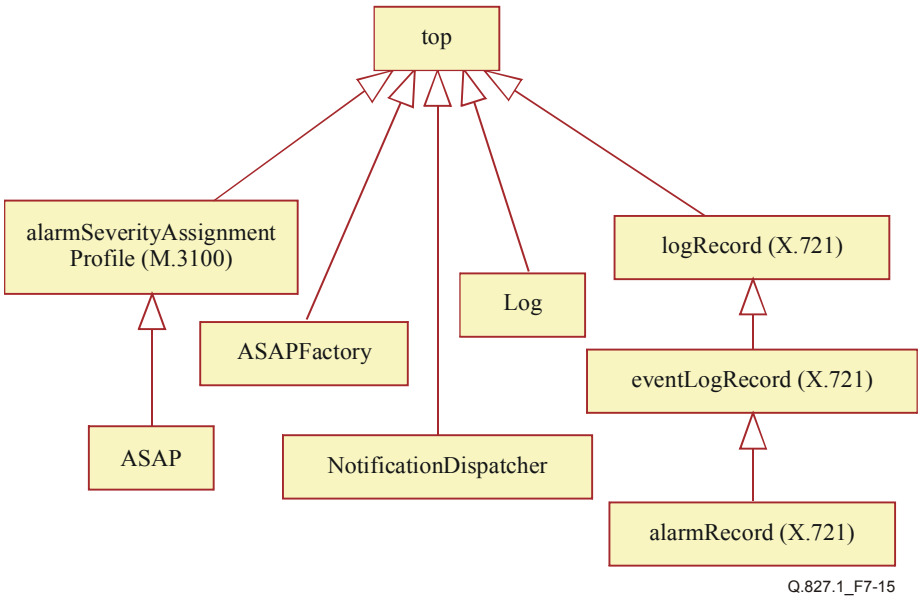

Figure 7-15/Q.827.1 – Inheritance diagram of the fault management

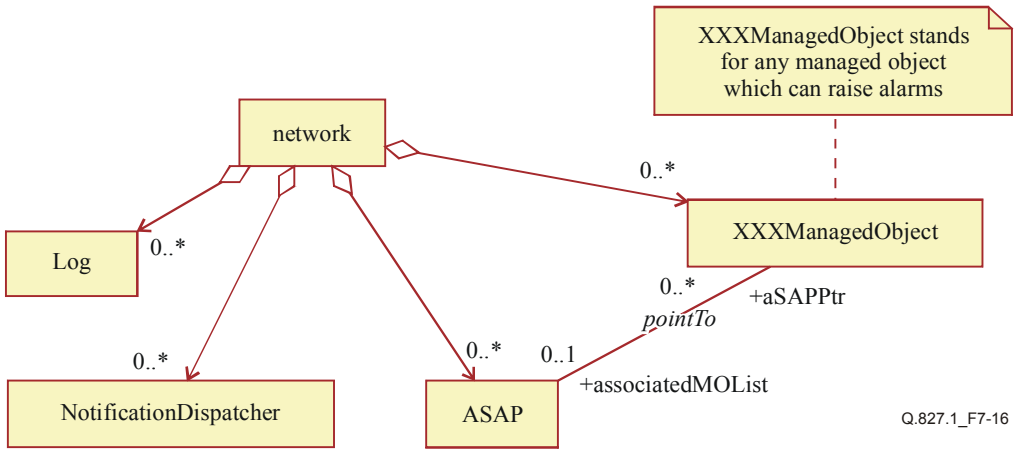

Figure 7-16/Q.827.1 – Entities relationship diagram of fault management

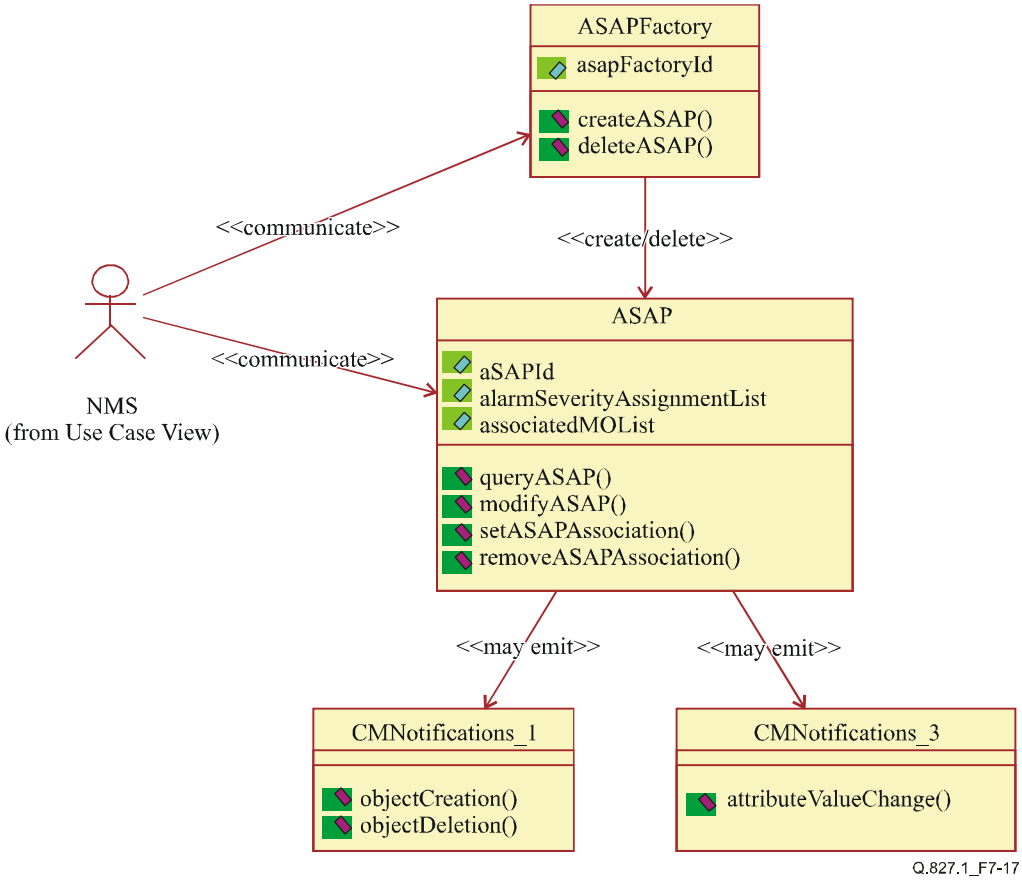

Figure 7-17/Q.827.1 – Class diagram of ASAP

7.4.1.2 ASAPFactory

Behaviour: This managed entity is used to create or delete ASAP instances.			
Attributes			
Name	**Description**	**Type**	**Qualifier**
factoryId	This is the unique identifier of the managed entity.	Integer	M, R
Operations			
Name	Description		
createASAP	This operation is used to create an ASAP instance.		
deleteASAP	This operation is used to delete an ASAP instance.		
Relationships: Zero or one instance of this managed entity exists in an instance of derived class of network.			

7.4.1.3 ASAP

Behaviour: This managed entity is derived from alarmSeverityAssignmentProfile defined in ITU-T Rec. M.3100. It specifies the alarm severity assignment for managed objects. Instances of this managed entity are referenced by the alarmSeverityAssignmentProfilePtr attribute in different managed objects.

Attributes			
Name	**Description**	**Type**	**Qualifier**
aSAPId	This is the unique identifier of the managed entity.	Integer	M, R

alarmSeverityAssignment List	The Alarm Severity Assignment List is an attribute type whose value provides a listing of all abnormal conditions that may exist in instances of an object class, and shows the assigned alarm severity information (minor, major, etc.) for each condition.
	The proposed type is listed as follows:
	AlarmSeverityAssignmentListType ::= LIST of STRUCT{
	problem: ProbableCause,
	severityAssignedServiceAffecting: AlarmSeverityCode (optional);
	severityAssignedNonServiceAffecting:
	severityAssignedServiceIndependent: AlarmSeverityCode (optional);
	}
	AlarmSeverityCode ::= ENUM { non-alarmed, minor, major, critical, warning }
		Alarm Severity Assignment ListType (LIST of STRUCT)
		See the left column for details.
			M, R/W

associatedMOList	This attribute specifies the MO instances this ASAP is associated with.	LIST of Name	M, R/S
Operations			
Name	Description		
queryASAP	This function is used to query the attribute values of this ASAP instance.		
modifyASAP	This function is used to modify the "alarmSeverityAssignmentList" attribute value.		
setASAPAssociation	This function is used to set one or more MO instances associated with this ASAP instance.		
removeASAPAssociation	This function is used to remove the association between this ASAP instance and one or more MO instances that are associated with it.		
Relationships: Zero or more instances of this managed entity exist in an instance of derived class of network.			
Reportable Notifications			
objectCreation			O
objectDeletion			O
attributeValueChange			O

7.4.2 ASAP management function set

7.4.2.1 Sequence diagram

Figure 7-18 illustrates the sequence of ASAP management.

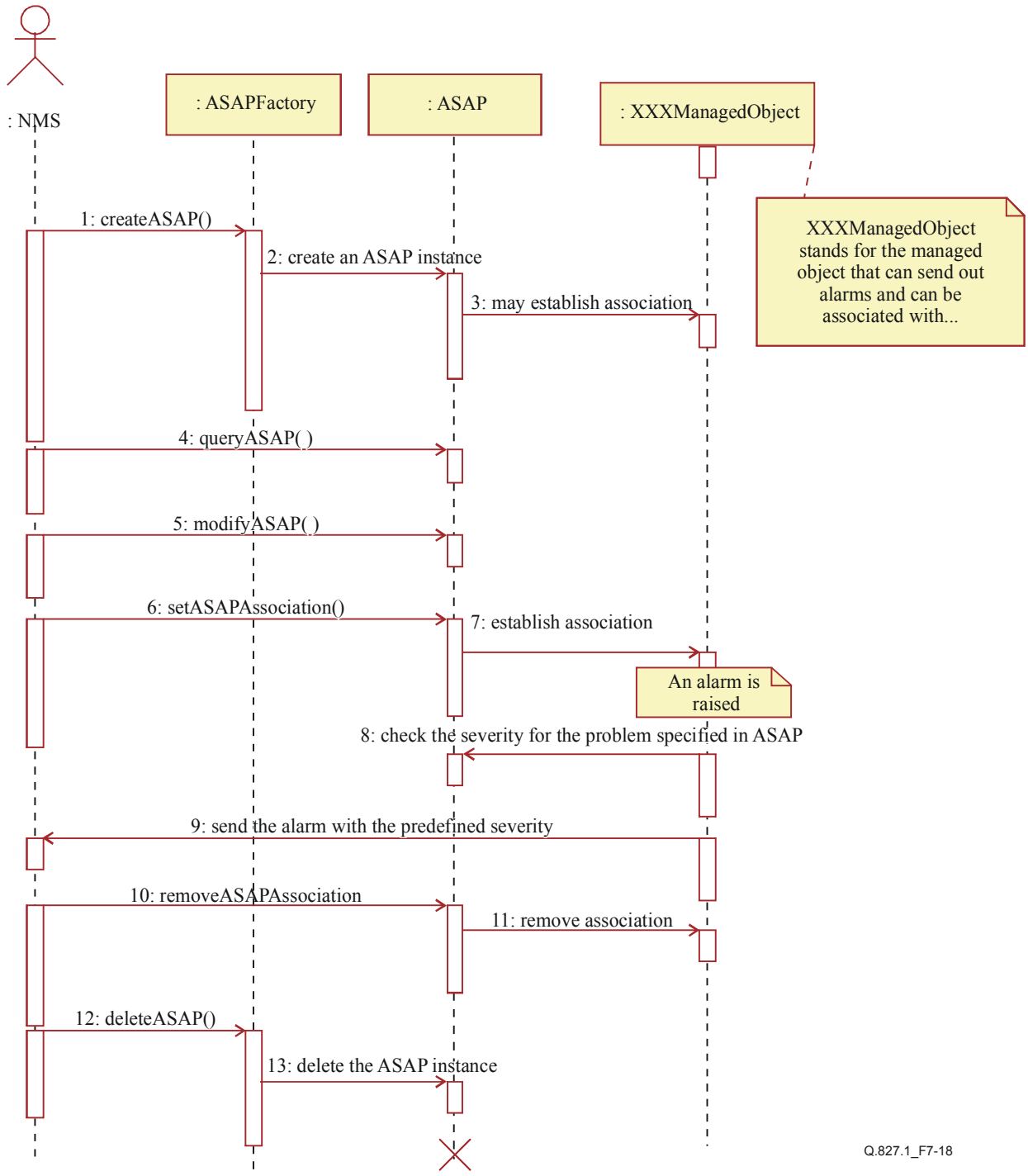

Figure 7-18/Q.827.1 – Sequence diagram of ASAP management functions

7.4.2.2 Management operations

1) "createASAP" Operation

Owner Entity	ASAPFactory
Description	This function is used to create an ASAP object instance. The input parameters are a list of the problems and their corresponding severity, and a list of the managed entities to be associated with them. If the creation succeeds, EMS will return success information, and may send an object creation notification. If the creation fails, EMS will send error information back to NMS.

Operation fields	Name	Description	Type
Input parameters	alarmSeverityAssignment List	This parameter is a list of the problems and their corresponding severity.	LIST of STRUCT { problem: Probable Cause; severity: ENUM; }
	associatedMOList	This attribute specifies the MO instances to be associated with this ASAP. This parameter can be empty when creating an ASAP instance.	LIST of Name
Output parameters	aSAPId	This parameter specifies the unique ID of this ASAP instance.	Name
Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	At least one of the input parameters is invalid.	
	UnknownManagedEntity	One or more managed entities specified in the request are unknown to EMS.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

2) "deleteASAP" Operation

Owner Entity	ASAPFactory		
Description	This function is used to delete an ASAP. The input parameter is the identifier of the ASAP instance. The ASAP to be deleted should not be associated with any managed objects; otherwise it cannot be deleted. If the deletion succeeds, EMS returns success indication, and may send an object deletion notification to NMS. If the deletion fails, EMS will send error information back to NMS.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	aSAPId	This parameter specifies ASAP managed objects to be deleted.	Name
Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	UnknownASAP	The specified ASAP object is unknown to EMS.	
	ASAPAssociationNot Removed	Association related with the ASAP object is not removed yet.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

3) "modifyASAP" Operation

Owner Entity	ASAP
Description	This function is used to modify the attribute values of an ASAP object. The input parameter contains the list of the problems and their corresponding severity can be modified. If the modification succeeds, EMS will send success information. If the modification fails, EMS will send error information back to NMS.

Operation fields	Name	Description	Type

Input parameters	alarmSeverityAssignment List	This parameter is a list of problems and their corresponding severity, which will be used to replace the old value of the alarmSeverityAssignmentList attribute of the ASAP.	LIST of STRUCT {
			problem: Probable Cause;
			severity: ENUM;
			}

Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	InvalidParameter	At least one of the input parameters is invalid.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

4) "queryASAP" Operation

Owner Entity	ASAP		
Description	This function is used to query the attribute information of an ASAP. If the operation succeeds, EMS will return the corresponding attribute values of the ASAP. If the operation fails, EMS will return error information.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	–	–	–

Output parameters	alarmSeverityAssignment List	This parameter is a list of the problems and their corresponding severity.	LIST of STRUCT {
			problem: Probable Cause;
			severity: ENUM;
			}

	associatedMOList	This parameter specifies the MO instances associated with this ASAP.	LIST of Name
Return Value	–	Success indication	Boolean
Exceptions raised	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

5) "setASAPAssociation" Operation

Owner Entity	ASAP
Description	This function is used to associate the ASAP object with one or more managed objects. The "associatedMOList" attribute of this ASAP will be automatically modified accordingly. The input parameter is a list of IDs of the managed objects to be associated with the ASAP. If the operation succeeds, EMS will return success information, and the ASAP starts to take effect on the specified managed objects. If the operation fails, EMS will return error information.

Operation fields	Name	Description	Type
Input parameters	moInstanceList	This parameter specifies the MO instances to be associated with the ASAP object.	LIST of Name
Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	UnknownManagedEntity	The specified MO instance(s) for association is unknown to EMS.	
	AssociationAlreadyExist	Association between ASAP object and the MO instance(s) already exists.	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

6) "removeASAPAssociation" Operation

Owner Entity	ASAP		
Description	This function is used to remove the association between the ASAP object and some of its associated managed object(s). The input parameter is a list of IDs of the managed objects associated with the ASAP. If the operation succeeds, EMS will return success information, and the ASAP will not take effect for the specified object(s) any longer. If the operation fails, EMS will return error information to NMS.		
Operation fields	**Name**	**Description**	**Type**
Input parameters	moInstanceList	This parameter specifies the managed objects whose association with the ASAP object will be removed.	LIST of Name
Output parameters	–	–	–
Return Value	–	Success indication	Boolean
Exceptions raised	UnknownManagedEntity	The specified managed object(s) is unknown to EMS.	
	AssociationNotExist	No association exists between the ASAP and the specified managed object(s).	
	EMSProcessingError	Error occurs during EMS processing.	
	CommunicationError	Communication error occurs.	

7.4.3 Related notifications

1) objectCreation
2) objectDeletion
3) attributeValueChange
4) equipmentAlarm
5) environmentalAlarm
6) communicationAlarm
7) processingErrorAlarm
8) qualityOfServiceAlarm

7.5 Security management function set

For further study.

Appendix I

Table of managed entities

Managed entity name in this Recommendation	Statement
NotificationDispatcher	Newly defined. Related to eventForwardingDiscriminator in ITU-T Rec. X.721.
NotificationDispatcherFactory	Newly defined.
Log	Newly defined. Related to log defined in ITU-T Rec. X.721.
LogFactory	Newly Defined.
logRecord (ITU-T Rec. X.721)	Defined in ITU-T Rec. X.721.
eventLogRecord (ITU-T Rec. X.721)	Defined in ITU-T Rec. X.721.
alarmRecord (ITU-T Rec. X.721)	Defined in ITU-T Rec. X.721.
attributeValueChangeRecord (ITU-T Rec. X.721)	Defined in ITU-T Rec. X.721.
stateChangeRecord (ITU-T Rec. X.721)	Defined in ITU-T Rec. X.721.
objectCreationRecord (ITU-T Rec. X.721)	Defined in ITU-T Rec. X.721.
objectDeletionRecord (ITU-T Rec. X.721)	Defined in ITU-T Rec. X.721.
BulkDataTransferReadyRecord	Newly defined.
BulkDataTransferPreparationErrorRecord	Newly defined.
FileTransferController	Newly defined.
network (ITU-T Rec. M.3100)	Defined in ITU-T Rec. M.3100.
MeasurementJob	Newly defined.
MeasurementJobFactory	Newly defined.
ThresholdMonitor	Newly defined.
ThresholdMonitorFactory	Newly defined.
ASAP	Newly defined. Related to alarmSeverityAssignmentProfile in ITU-T Rec. M.3100.
ASAPFactory	Newly defined.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	**Switching and signalling**
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure, Internet protocol aspects and Next Generation Networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2005