

January 11, 2001

T1M1

Internetwork Operations, Administration,
Maintenance, and Provisioning

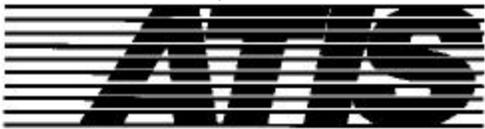
A Technical Subcommittee of
Standards Committee T1
Telecommunications
www.t1.org

Accredited by the American National
Standards Institute

Michael J. Fargano
Chairman

James T. Stanco
Vice Chairman

A Sponsored Committee of



Alliance for Telecommunications
Industry Solutions

Michael J. Fargano
Qwest Communications
4001 Discovery Dr.
Boulder, CO 80303
303 541 6081 (T)
303 541 8344 (F)
email: mfargan@qwest.com

Albert H. Yuhan
3GPP TSG-SA/WG5 Chairman
ayuhan@omnipoint-pcs.com

Subject: Coordination of 3GPP and T1M1 CORBA Based Fault Management (Alarm Surveillance) and Performance Monitoring Work Efforts

Dear Albert,

T1M1.5 has started the development of IDL specifications in support of Performance Monitoring (PM) and Fault Management (FM) functions; for FM, specifically the Alarm Surveillance function. These documents are being developed using as the basis ITU Rec. Q.822, Q.821 and the recently developed CORBA framework (Q.816 and X.780). These documents will be submitted to ITU SG 4 meeting in January 2001. The schedule for the two documents is to issue them for ballot (if a T1 standard is required) during the February 26, 2001 T1M1 meeting week or progress them through the new ITU procedure. In either case, stable documents are expected to be available by September 2001.

We understand that 3GPP SA5 has started a work item for PM IRP (Interface Reference Point), which includes a CORBA Based solution set.

The T1M1 proposed FM and PM functions provide for key common CORBA based OAM&P components that are required across multiple network technologies. This common approach provides for significant standards/industry consistency and efficiency.

In order to avoid the repetition of the models in two different groups, T1M1 proposes that the attached models be discussed and that 3GPP provide T1M1 with comments. Such an early interaction will facilitate T1M1 taking into account 3GPP concerns before issuing the documents for ballot.

The models are complete except for the open issue discussed in the PM specification. This relates to the use of FTP to support transferring the PM data, which can be large.

Please contact me if you have any questions, comments, of concerns.

Thank you,

Mike Fargano
T1M1 Chairman

Attachments:
T1M1.5/2000-286R1
T1M1.5/2000-217R3

Cc (w/ Attachments):

Michael Truss, Vice Chair 3GPP TSG-SA WG5
Masaaki Yoshimura, Chair 3GPP2 TSG-S WG2
Jörg Schmidt, 3GPP2 to T1M1 Liaison
Athan Tuzel, Chair ATM Forum NM Working Group

Cc (w/o Attachments):

Niels Andersen, Chair 3GPP TSG-SA
Gary Jones, Vice Chair 3GPP TSG-SA
Armin Toepfer, Vice Chair 3GPP TSG-SA
Hideo Okinaka, Chair 3GPP2 TSG-S
Cheryl Blum, Chair TIA TR45
Asok Chatterjee, Chair T1P1
Mark Younge, Vice Chair T1P1
Al Vincent, Technology Director TMF
Rick Townsend, Chair ATM Forum
Gerhard Maegerl, Vice Chair ATM Forum
T1M1 Leadership Team

**COMMITTEE T1 – TELECOMMUNICATIONS
Working Group T1M1.5 and T1M1.3 Meeting
Columbus, OH; December 6, 2000**

**T1M1.5/2000-286r1
T1M1.3/2000-140r1**

DRAFT STANDARD

Title: Working Document for Draft Standard ANSI T1.2xx-2001,
CORBA-based Performance Management Information Model

Source: Editors

Contact: Bing Leng Weijing Chen Trevor Pirt
Lucent Technologies SBC Motorola
(630) 713-8333 (512) 372-5710 (353) 21-357101
bleng@lucent.com wchen@tri.sbc.com trevor.pirt@motorola.com

Distribution: T1M1.5, T1M1.3

Project: Protocol Standards for Communication between Operating Systems

ABSTRACT

This document defines an information model to be used in telecommunications performance management (PM) based on CORBA. It defines in Interface Definition Language (IDL) a set of interfaces, notifications, and constants. The intent of this document is to define a CORBA/IDL model similar to that defined in ITU Recommendations X.739 and Q.822 using CMISE. This document is compliant with proposed CORBA modeling standards Recommendation X.780 and Recommendation M.3120. This version of the document is based on T1M1.5/2000-286 and the agreement of the 11-2000 Orlando meeting.

NOTICE

This is a draft document and thus, is dynamic in nature. It does not reflect a consensus of Committee T1-Telecommunications and it may be changed or modified. Neither ATIS nor Committee T1 makes any representation or warranty, express or implied, with respect to the sufficiency, accuracy or utility of the information or opinion contained or reflected in the material utilized. ATIS and Committee T1 further expressly advise that any use of or reliance upon the material in question is at your risk and neither ATIS nor Committee T1 shall be liable for any damage or injury, of whatever nature, incurred by any person arising out of any utilization of the material. It is possible that this material will at some future date be included in a copyrighted work by ATIS.

ANSI[®]
T1.2XX-2001

**American National Standard
for Telecommunications**

CORBA-based Performance Management Information Model

Secretariat
Alliance for Telecommunications Industry Solutions

Approved XXX 31, 2001
American National Standards Institute

Abstract

This document defines an information model to be used in telecommunications performance management (PM) based on CORBA. It defines in Interface Definition Language (IDL) a set of interfaces, notifications, and constants. The intent of this document is to define a CORBA/IDL model similar to that defined in ITU Recommendations X.739 and Q.822 using CMISE. This document is compliant with proposed CORBA modeling standards Recommendation X.780 and Recommendation M.3120.

Table Of Contents

TABLE OF FIGURES	VI
TABLE OF TABLES	VI
FOREWORD	VII
1. INTRODUCTION	9
2. REFERENCES	10
3. TERMS AND DEFINITIONS	11
4. ACRONYMS	11
5. OVERVIEW OF THE PM INFORMATION MODEL	12
5.1 REQUIREMENTS.....	12
5.1.1 <i>Functional Requirements</i>	12
5.1.2 <i>Modeling Requirements</i>	13
5.2 INFORMATION MODEL.....	13
5.2.1 <i>Scope</i>	13
5.2.2 <i>UML Model</i>	15
5.2.3 <i>Containment Tree</i>	18
5.3 INFORMATION MODEL DESCRIPTION	18
5.3.1 <i>Scanner</i>	18
5.3.2 <i>CurrentData</i>	19
5.3.3 <i>ZsCurrentData</i>	21
5.3.4 <i>HistoryData</i>	21
5.3.5 <i>ThreshodData</i>	22
5.3.6 <i>HistoryDataScanner</i>	22
5.4 DOCUMENTATION CONVENTION OF THE MODEL.....	25
6. PM INFORMATION MODEL IDL	26
6.1 IMPORTS.....	26
6.2 FORWARD DECLARATIONS.....	27
6.3 STRUCTURES AND TYPEDEFS.....	28
6.4 EXCEPTIONS.....	31
6.4.1 <i>Exceptions for Conditional Package</i>	31
6.4.2 <i>Exceptions for Performance Management</i>	31
6.5 INTERFACES.....	32
6.5.1 <i>Scanner</i>	32
6.5.2 <i>CurrentData</i>	35
6.5.3 <i>ZsCurrentData</i>	39
6.5.4 <i>HistoryData (ValueType)</i>	41
6.5.5 <i>ThresholdData</i>	43
6.5.6 <i>HistoryDataScanner</i>	46
6.6 NOTIFICATIONS.....	47
6.7 NAME BINDING.....	48
6.7.1 <i>ThresholdData</i>	48
7. OPEN ISSUES	49

Table of Figures

Figure 1. Use Cases for Performance Measurement.....	13
Figure 2. Class Diagram for CurrentData, HistoryData, and ThresholdData	15
Figure 3. Scanner.....	16
Figure 4. History Data Scanner.....	17
Figure 5. Containment Tree	18

Table of Tables

Foreword

This document was produced by the T1M1.5 Working Group on OAM&P Architecture, Interface, and Protocols.

CORBA, a distributed processing technology, has been accepted by ITU-T for X interface as well as Q interface in Telecommunications Management Network (TMN). A CORBA-based telecommunications network management framework and a generic network information model have been defined by T1M1.5 and have been submitted to ITU-T for standardization. The framework and the generic information model provide basis for further modeling in TMN functional areas. One of the key TMN functional areas is performance management that deals with data collection, data storage, thresholding, and data reporting for the purpose of achieving high Quality of Service in the network. This document defines the performance management information model, which will be submitted to the ITU-T in an attempt to develop international standards for CORBA-based performance management interfaces.

Suggestions for the improvement of this standard are welcome. They should be sent to the Alliance for Telecommunications Industry Solutions, 1200 G Street, NW, Suite 500, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by the Accredited Standards Committee on Telecommunications T1. Committee approval of this standard does not necessarily imply that all committee members voted for its approval.

The following members of the T1M1.5 Management Services Sub-Working Group contributed to the effort to produce this document:

Lakshmi Rahman, <i>Chair</i>	Chris Kang	Tom Rutt
Weijing Chen, <i>Editor</i>	H. Kam Lam	Randy Scheer
Bing Leng, <i>Editor</i>	Dave Matthews	Jim Stanco
Trevor Pirt, <i>Editor</i>	Bernie Mayer	Wendy Teller
Keith Allen	Rick Ordower	John Wilber
Albert Bonavolonta	Tom Plevyak	
Tom Grim	John Portschy	

American National Standard for Telecommunications

CORBA-based Performance Management Information Model

1. Introduction

This document defines an information model to be used in telecommunications performance management (PM) based on CORBA. It defines in Interface Definition Language (IDL) a set of interfaces, notifications, and constants. The intent of this document is to define a CORBA/IDL model similar to that defined in ITU Recommendations X.739 and Q.822 using CMISE. This document is compliant with proposed CORBA modeling standards Recommendation X.780 and Recommendation M.3120.

This document has the following sections:

- Section 1. Introduction
- Section 2. References
- Section 3. Terms and Definitions
- Section 4. Acronyms
- Section 5. Overview of the PM Information Model
- Section 6. PM Information Model IDL
- Section 7. Open Issues

2. References

This section contains references for documents on which this specification draws.

- [1] The Object Management Group (OMG), "The Common Object Request Broker: Architecture and Specification", OMG Document formal/99-10-07, Revision 2.3.1, October 1999.
- [2] The Object Management Group (OMG), "CORBA Services: Common Object Services Specification", Updated version, December 1998.
- [3] The Object Management Group (OMG), "Notification Service", OMG TC Document telecom/98-11-01, November 3, 1998.
- [4] ITU-T Draft Recommendation, Q.816, "*CORBA-Based TMN Services*".
- [5] ITU-T Draft Recommendation, X.780, "*TMN Guidelines for Defining CORBA Managed Objects*".
- [6] ITU-T Recommendation, X.739 (1993), "*Information technology – Open Systems Interconnection – Systems management: Metric objects and attributes*".
- [7] ITU-T Recommendation, X.738 (1993), "*Information technology – Open Systems Interconnection – Systems management: Summarization function.*".
- [8] ITU-T Recommendation Q.822, (1994), "*Stage 1, Stage 2 and Stage 3 description for the Q3 interface – Performance management*".
- [9] ITU-T Recommendation, X.792 (1999), "*Configuration audit support function for ITU-T applications*".
- [10] T1M1.5/2000-029, "*Framework for CORBA-based Telecommunications Management Network Interfaces*".
- [11] T1M1.5/2000-030, "*CORBA Generic Network and NE Level Information Model*".
- [12] T1M1.5/2000-217, "*Proposal for New Addendum to Recommendation Q.821*".

3. Terms and Definitions

This document use the terms and definitions defined in X.739, Q.822, T1M1.5/2000-029, and T1M1.5/2000-030.

4. Acronyms

The following terms and acronyms are defined in X.739 or Q.822 and used in this document.

NE	Network Element
PM	Performance Management
QoS	Quality of Service
TMN	Telecommunications Management Network

5. Overview of the PM Information Model

5.1 Requirements

5.1.1 Functional Requirements

Performance Measurement is used in two ways in TMN. One way is in the measure of the performance of transport and protocol entities. In this use the measurements are subjected to thresholding and are collected for engineering and service history on a time scale that is not real time critical. This is the use supported by the use of Q.822 and technology specific standards directly based on it. The other use of performance measurement is in support of network traffic management. In this application the measurements are not subjected to thresholding but are collected periodically on a time scale that is needed to support the application of network management controls. The typical time scale for this collection has historically been 5 minutes. This is the use supported by Q.823. Q.823 reuses the mechanisms of Q.822 together with the generic simple scanner of X.738 to produce a single scan report that summarized the previous granularity period and is reported to a network management OS. These two uses of performance measurement are summarized in the Use Cases shown in Figure 1.

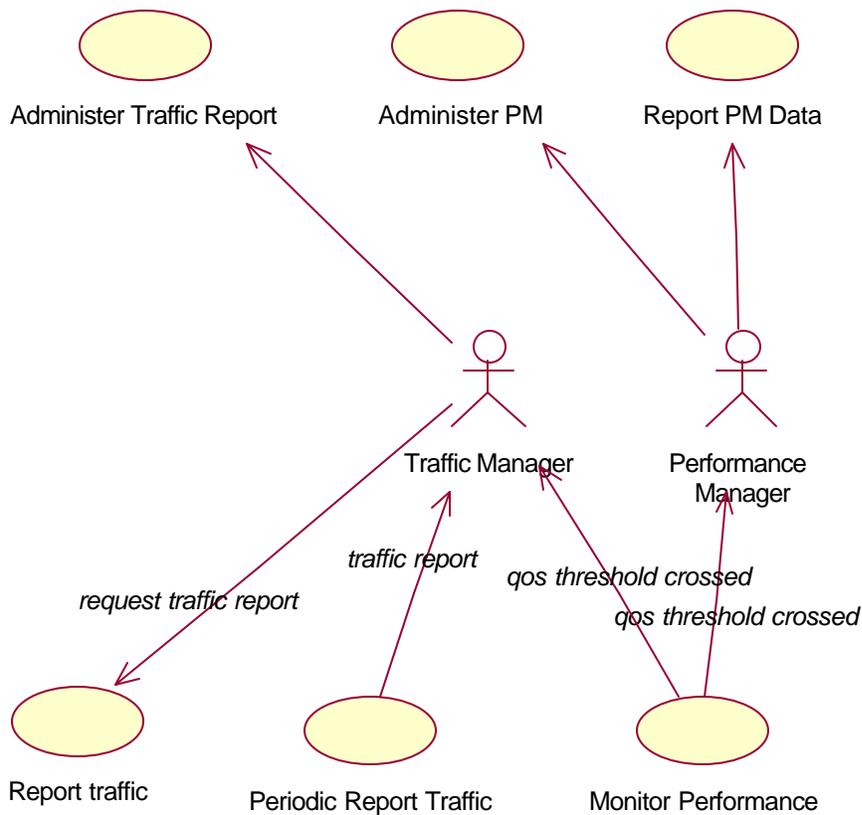


Figure 1. Use Cases for Performance Measurement

In summary, the TMN Management functions for performance management include data collection, data storage, thresholding, and data reporting.

Performance data collection refers to the ability for a NE to collect the various PM data relating to a single monitored entity in that NE. This function allows a TMN manager to assign PM data collection interval, to suspend/resume PM data collection process, and to reset the performance monitoring counters.

PM data storage refers to the capability for a NE to store historical PM data on each monitored entity for prescribed time duration. This function allows a TMN manager to establish a duration during which to maintain a specific record of PM historical data, to optionally screen historical data based on some criteria (e.g., suppress "all-zero" data), and to remove historical PM data at the end of time interval.

PM thresholding refers to the ability for a NE to inform a TMN manager of any threshold crossing. It also provides the TMN manager with the means for establishing thresholding criteria.

PM data reporting refers to the capability for a NE to report PM data on a scheduled basis, or as a result of a spontaneous request from the TMN manager. A report may contain data from a given monitored entity, or it can contain summarized data from a set of monitored entities. This function allows a TMN manager to request PM data, to allow/inhibits the emission of scheduled reports in the NE, and to screen the PM data reports on some criteria (e.g., suppress "all-zero" data).

5.1.2 Modeling Requirements

This document follows the information modeling guidelines defined in T1M1.5/2000-029. In addition, this document tries to keep the model practical, simple and make sense to the users and products.

5.2 Information Model

5.2.1 Scope

To support the functional requirements the performance management model needs `currentData` to collect performance measurements, `historyData` to store the collected measurements, `thresholdData` to specify the thresholds, and some specific scanner to report history data. Since `currentData` is inherited from `Scanner`, the model also needs `Scanner`.

However, based on the modeling guidelines and the needs from existing and current PM applications, not all the capabilities defined for these managed objects in X.739 and Q.822 are necessary at this point in time. As a result, some of the optional packages are not included in this model. In the future, if the need for a capability arises, the relevant package(s) will be reconsidered for inclusion.

For most managed objects identified above, their IDL model can be derived from a translation of the X.739 and Q.822 GDMO. For historyData, however, direct translation may result in potentially large number of managed objects. These entities are read only data stores of historical data and do not need to be CORBA interfaces. Instead it is proposed that they are to be defined as CORBA valuetype objects. The Q.822 translation will define a base class of historyData that is a valuetype object. These valuetype history data objects will be accessible through methods on the currentData interface. When current data is subclass it is intended that a history data valuetype be subclassed along with it.

Since the data that needs to be stored is not in CORBA interfaces but in history data valuetypes, to report history data, simply translating the GDMO homogeneous and simple scanners will not work. Instead, a new kind of scanner is needed - one that collects data from the history data valuetypes. This model is attempting to keep the interfaces strongly typed whenever possible. This is because the collection of traffic data is computationally intensive since large amount of data must be collected and processed by the network manager in a short period of time.

For this purpose, HistoryDataScanner is designed to retrieve history data saved under the relevant currentData objects. Two kinds of operations are necessary for a managing system to retrieve data - notification and file transfer.

Notification is used when a traffic control is issued and the managing system needs to know the effectiveness of the control. In this situation, the amount of monitoring data is small but the feedback needs to be quick, which makes the notification a good choice. For now, notification is still a placeholder, the detailed definition will be provided in the further version of the document.

File transfer mechanism is used for retrieving large amount of data since other techniques are likely to be both slower and to consume more computing and communication resources. To support file transfer, a file format and a set of operations are defined. However, the name and location of the file, the way the file is to be transferred, and the duration and deletion of the file are outside the scope of this document.

The file transfer mechanism used in the document is based on the technique defined in X.792. Other approach might need to be investigated such as OMG ftp/ftam file transfer specification.

The resulting model described in UML is shown in the next section.

5.2.2 UML Model

This section illustrates the UML model for performance management. The model covers Scanner, CurrentData, HistoryData, ThresholdData, and HistoryDataScanner. Since the model can not be fit into one page, it is displayed in the next three figures.

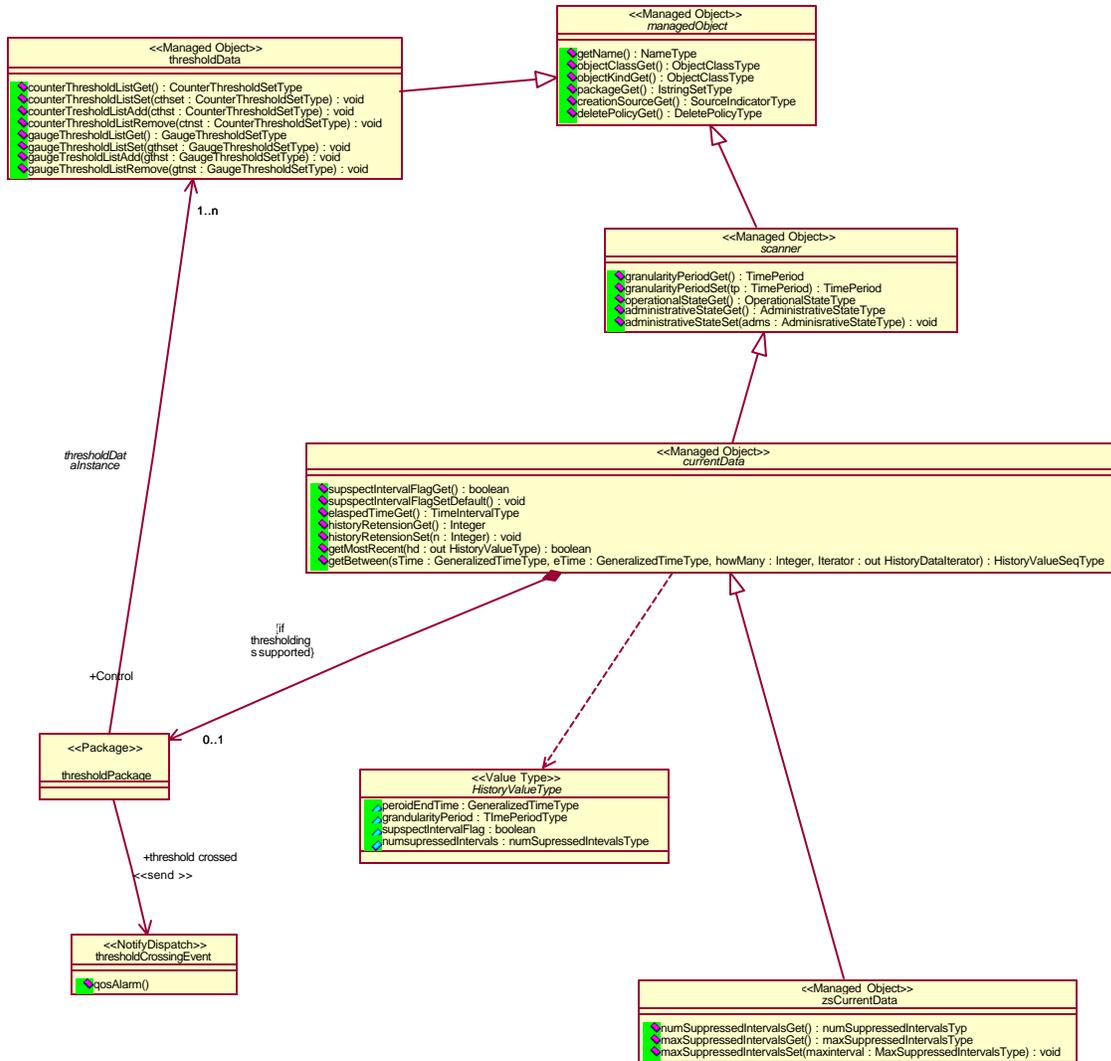


Figure 2. Class Diagram for CurrentData, HistoryData, and ThresholdData

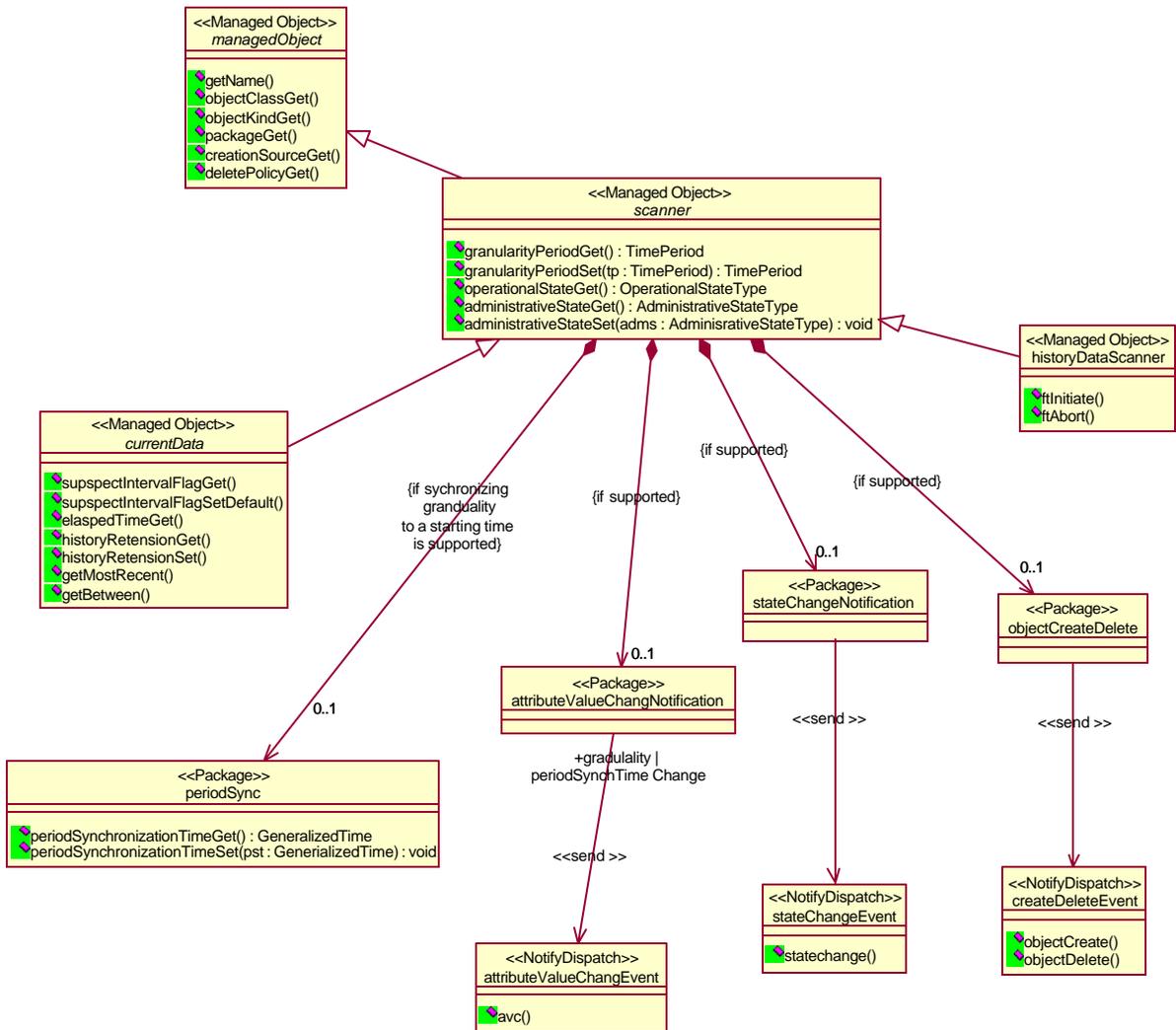


Figure 3. Scanner

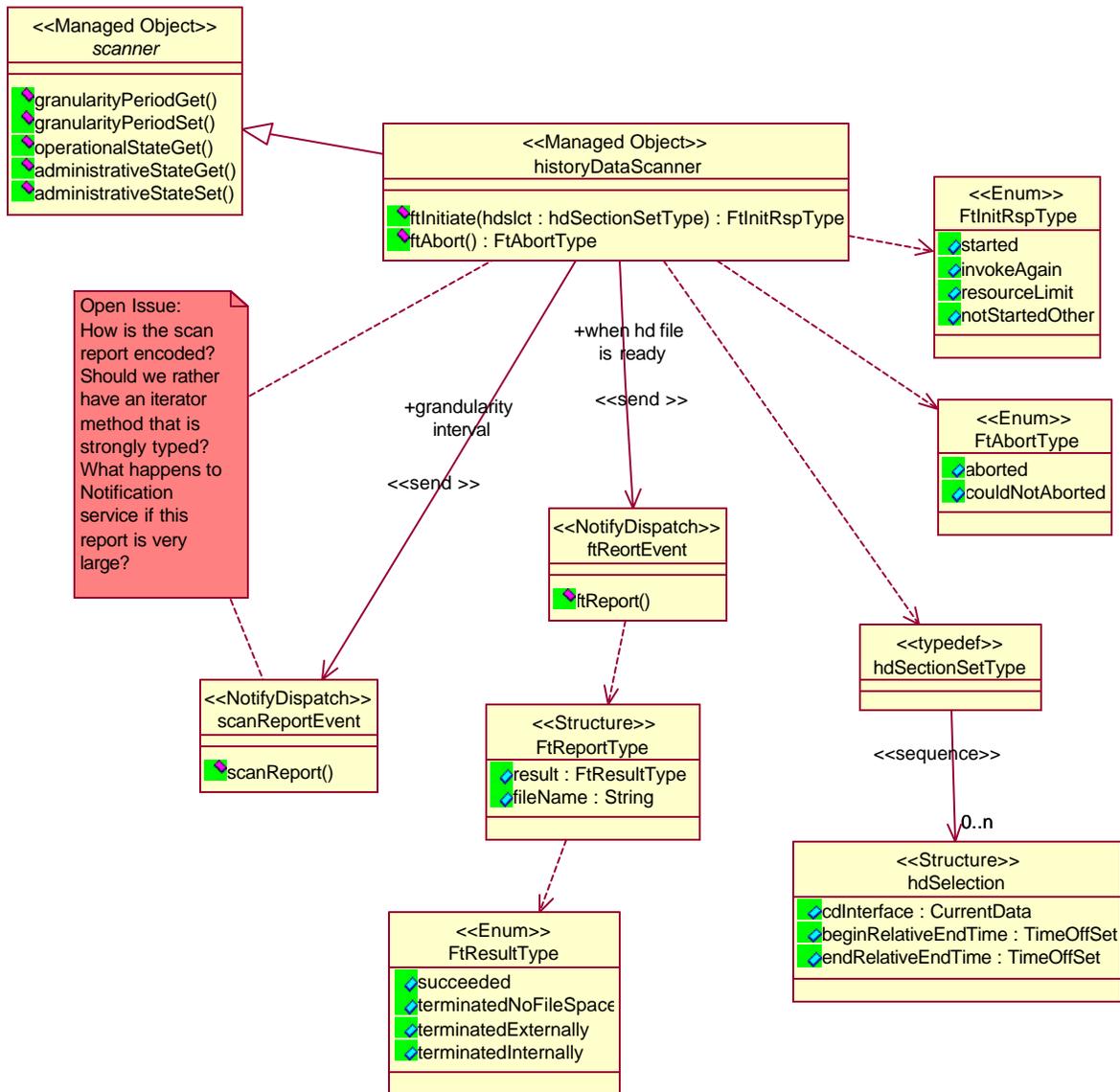


Figure 4. History Data Scanner

5.2.3 Containment Tree

This section illustrates the containment relationship of the CORBA interfaces defined in this document.

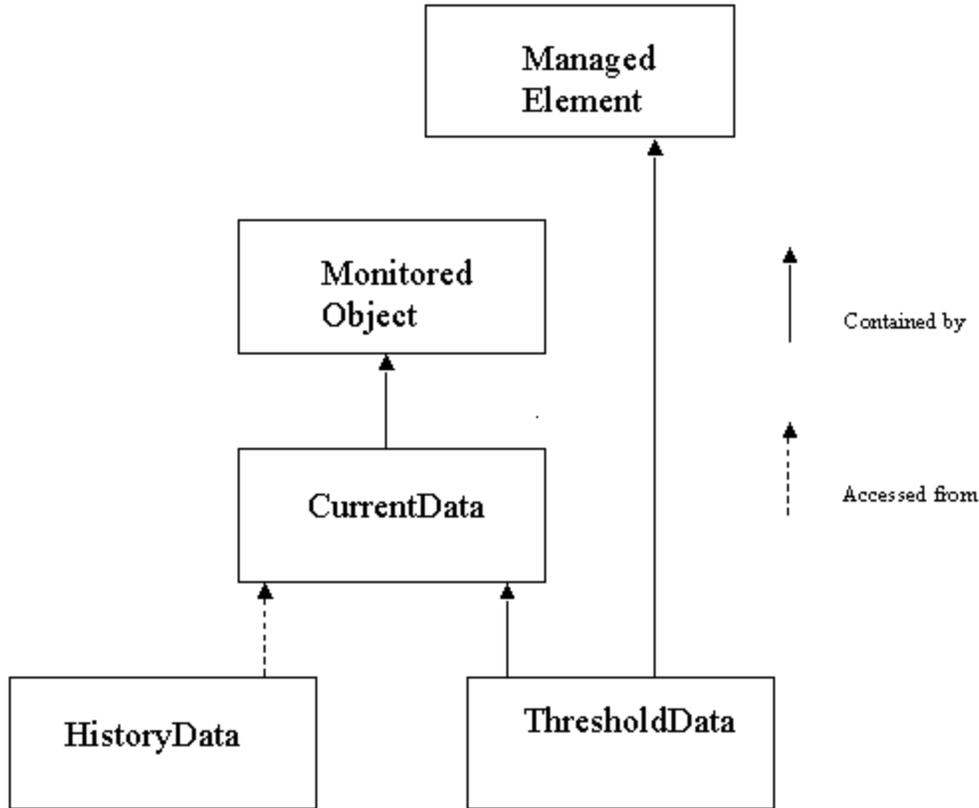


Figure 5. Containment Tree

5.3 Information Model Description

5.3.1 Scanner

A managed object of this interface represents the ability to retrieve values of attributes of managed objects and produce summary information from those values. This summary information may be made available in attributes, notifications, action replies, or some combination of these. Summary information may consist of observed attribute values or statistics calculated from these values (either over time or over managed objects).

Observed attribute values are retrieved during a "scan", which is initiated periodically, at the end of each granularity period, provided that the granularity period is non-zero.

The granularity period attribute indicates the length of the granularity period. The granularity period in the scanner object shall not be modified unless the value of the administrative state is 'locked'. If the period synchronization package is not present, the time at which a granularity period starts after the scanner is unlocked is a local matter.

The administrative state attribute is used to suspend or resume the scanning function. If administrative state has the value 'unlocked', the scanner is administratively permitted to perform scans. If administrative state has the value 'locked', the scanner is administratively prohibited from performing scans.

The operational state attribute represents the operational capability of the scanner to perform its functions.

If the attribute value change notification package is present, then changes to the granularity period shall cause attribute value change notifications to be emitted. If the state change notification package is present then changes to the operational state or administrative state shall cause state change notifications to be emitted. If the object create/delete notification package is present, then the creation or deletion of an object of scanner's subclass shall emit the corresponding create or delete notification.

The Scanner interface is not instantiable.

The following attributes are defined in the entity:

- *administrativeState* - is used to activate and deactivate (administratively suspend and resume) the function performed by the scanner.
- *granularityPeriod* - specifies the length of time during which the "scan" is performed.
- *operationalState* - identifies whether or not the scanning function represented by this object is capable of performing its normal functions.
- *periodSynchronizationTime* [optional] - contains the time to which a repeating time period is synchronized. The start of each time period is an integral number of periods before or after the time specified by this attribute.

5.3.2 CurrentData

The currentData object class, a particular type of scanner, is a class of managed support objects that record the current performance data for monitoring purpose. The performance data (measurements) of monitored resources are modeled as attributes in the definition of subclasses of currentData since currentData can't be instantiated. The objects that represent monitored resources contain the corresponding currentData objects in the name binding relationship.

The measurements are collected for a time interval (e.g., 5 min.) specified by granularityTime attribute. At the end of each interval, the elapsedTime attribute will be updated to the difference between the current time and the start of the present monitoring interval. The measurements, granularityTime, and suspectIntervalFlag are copied into a

corresponding history data structure. Further more, the `periodEndTime` of the history data structure is assigned to the current time.

If `historyRetention` is greater than zero, then history data structures will be retained in the managed system for at least the duration equivalent to the number of intervals specified in the `historyRetention` attribute. During that time, history data can be accessed via a set of operations:

- `getMostRecent` - will retrieve the most recent history data
- `getBetween` - will retrieve a set of history data within a time window specified by start and end time points

If threshold package is present, the `currentData` object contains a pointer to a `thresholdData` object. If any of the thresholds (defined in the referenced `thresholdData` object) are violated a Quality of Service (QoS) alarm notification is emitted by the `currentData` object. The `thresholdInfo` field of the alarm shall contain all the measurement attributes. If additional threshold acrossing(s) occur during the current time interval then additional alarms shall be emitted as well.

New thresholds resulting from modifying the `thresholdDataInstance` attribute or from changing a threshold value in the referenced `thresholdData` object, should take effect immediately. If an alarm condition exists previous to the occurrence of a threshold value change (i.e. an old threshold had been violated), and the new threshold value is outside of the range of the old threshold value (e.g., in the case of an increasing counter, the new threshold value is greater than the old threshold value), and the current value of the measurement is within the allowable range of the new threshold value, then a QoS Alarm notification is emitted with a severity of 'clear'. If the new threshold value is set within the range of the old threshold value, such that the new threshold is violated, a QoS Alarm notification is emitted if an alarm condition is not already outstanding.

The `thresholdDataInstance` attribute contains a set of pointers to `thresholdData` objects, therefore, more than one QoS Alarm notifications might be emitted for a single monitored attribute. And it is possible the notifications might be conflict. For example, one notification indicates alarm is on, the other cleared. It is the managing systems' responsibility to maintain the notification consistency.

The following attributes are defined in the entity:

- *suspectIntervalFlag* - is used to indicate that the performance data for the current period may not be reliable. Some reasons for this to occur are:
 - Suspect data were detected by the actual resource doing data collection.
 - Transition of the `administrativeState` attribute to/from the 'lock' state.
 - Transition of the `operationalState` to/from the 'disabled' state.
 - The performance counters were reset during the interval.
 - The `currentData` (or subclass) object instance was created during the monitoring period.

- *elapsedTime* - represents the difference between the current time and the start of the present monitoring interval.
- *historyRetention* - specifies the minimum number of intervals that the history data structure (just being created) must be preserved.
- *thresholdDataInstance* - is a "list" attribute in which each item is a pointer to a thresholdData object that contains threshold limits for performance parameters. Whenever the value of a PM parameter violates its threshold setting, a qualityofServiceAlarm notification is emitted.

5.3.3 ZsCurrentData

ZsCurrentData is a CurrentData with zero suppression. Zero suppression is that when the measurements are all zeros no history data structures will be created.

The following attributes are defined in the entity:

- *numSuppressedIntervals* - is used to count the number of consecutive intervals for which suppression (i.e. non-creation of a history data structure) has occurred. This attribute reflects performance measurements up to, but not including, the current interval. This attribute gets incremented at the end of an interval if suppression has occurred. Otherwise, the attribute is reset. Or it reaches maxSuppressedIntervals, a history data record will be created and itself will be reset.
- *maxSuppressedIntervals* - limits the maximum number of suppressed intervals that will be collected without creating a structure of the history data.

5.3.4 HistoryData

HistoryData object will contain a copy of the performance measurements and other selected attributes that are present in a current data object at the end of the current interval (e.g., 5 min.). A new record of this valuetype structure is created at the end of each interval if historyRetention attribute in the current data object is greater than zero and this record will be retained in the NE for at least the duration equivalent to the number of intervals specified in the historyRetention attribute.

This generic valuetype object models the history data. Specific current data objects (e.g., SDH/SONET current data objects) will be defined with specific history valuetype objects, which are inherited from this generic history valuetype object.

The following attributes are defined in the entity:

- *periodEndTime* - indicates the time at the end of the interval.
- *granularityPeriod* - is used to copy the same attribute from the currentData object.
- *suspectIntervalFlag* - is used to copy the same attribute from the currentData object.

- *numSupressedIntevals* - indicates the number of supressed intervals has occurred before this historyData. It has default value 0. When copying from currentData object, the default value is used. When copying from zsCurrentData object, the corresponding attribute value is used.

5.3.5 ThreshodData

The thresholdData object class is a class of managed support objects that contains the values of the threshold settings for the PM parameters. At least one of the counterThresholdListPackage or the gaugeThresholdListPackage must be instantiated.

Thresholds are established by use of the managed object class thresholdData. The thresholdData objects specify the thresholds being applied. Threshold values in a thresholdData object may apply to multiple currentData objects and thresholdData objects are pointed to by currentData objects. The currentData, or its subclasses, object indicates the collection interval used with the threshold. A qualityOfServiceAlarm notification is generated whenever a threshold is crossed.

A threshold may be specific to a single managed object. In this case, the thresholdData object is contained within the managed object being observed. A currentData object within the managed object points to a thresholdData object. This linkage is needed for consistency with the multiple managed object case.

It is sometimes desirable to have a threshold apply to a group of managed objects. In this case, the thresholdData object is external to the managed object being observed. It may be contained within the managedElement object. The thresholdData object is pointed to by all the currentData objects to which it applies.

If createDeleteNotificationsPackage is present, the creation or deletion of a threshold object will cause a objectCreation or objectDeletion notification be emitted.

If attributeValueChangeNotificationPackage is present, then any threshold change will cause an attributeValueChange notification being emitted.

The following attributes are defined in the entity:

- *counterThresholdList* [optional] - 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.
- *gaugeThresholdList* [optional] - contains a set of threshold settings for performance attributes of the gauge type. Each threshold setting consists of the attribute identifier, the threshold value and (optionally) the severity of the threshold-exceeded event.

5.3.6 HistoryDataScanner

HistoryDataScanner is a subclass of Scanner. It supports file transfer by providing a file transfer initiating action and a terminating action as well as a notification, which indicates the file, is ready.

The managing system requests the managed system to create a data file with the file transfer initiate request. The managing system may indicate in this request that only certain history data records be included in the file. The managed system determines if the requested data file can be created and returns this information to the managing system in its file transfer initiate response. The managed system then attempts to create the requested file. The results of the attempt to create the data file are reported by the managed system to the managing system in the file transfer report. If the file was successfully created, the manager may retrieve the file using a file transfer protocol.

The file transfer initiating action requests the initiation of the creation of a data file:

```
FtinitRspType ftInitiate(hdSectionSetType hdselection)
```

The parameter *hdselection* contains a set of following information:

- *current data interface* - indicates which current data object is considered
- *begin relative time* - the begin time relative to the current time together with the end relative time defining a window to select history data
- *end relative time* - the end time relative to the current time

The return value is an enum type with one of the followings:

- *started* - indicates the file is being created by the action invocation
- *invokeAgain* - the managed system is in the process of creating a data file and only one file can be created at a time, so the managing system needs to invoke the action again
- *resourceLimit* - indicates that the data file could not be created due to a resource limit in the managed system
- *notStartedOther* - indicates that the data file could not be created due to some other reason

File transfer terminating action is used to terminate an ongoing generation of a data file.

```
FtAbortType ftAbort()
```

It does not have any parameter. It returns the result of the attempted abortion. The following results are possible:

- *aborted* - indicates the data file generation has been stopped
- *could not aborted* - indicates that the data file generation could not be stopped

File transfer reporting notification is generated when the creation of the data file terminates. It is used to report the status of the termination as well as the name of the file if generated. The possible termination statuses are:

- *succeeded*

- *terminatedNoFileSpace*
- *terminatedExternally*
- *terminatedInternally*

The data file is created based on a file format, which is defined as follows:

Definition	Explanation
<p><HD File Format> := <Selection Criteria> <Separator> <Records>*</p>	<p>The history data file format consists of selection criteria and resulting history data records.</p> <p>The selection criteria indicate what this file is for and how the records are organized.</p> <p>The number and the order of current data objects specified in the selection criteria determine the number and the order of the corresponding history data records.</p> <p>There might be zero record if no current data object is specified in the selection criteria.</p>
<p><Records> := <Interface Name> < Separator> <Record>*</p>	<p>The records are the selected history data for a given current data object.</p> <p>The interface name of the current data object is listed here to indicate the data schema for the following records.</p> <p>There might or might not be a data record for the object.</p>
<p><Record> := (<Value> < Separator>)+</p>	<p>A history data record consists of measurement values. The exact number of values is determined by the corresponding history data structure.</p>
<p><Value> := string <Vary Sized Value></p>	<p>The fix-sized values are represented by string. The vary-sized values, such as set and sequence, are defined below.</p>
<p><Vary Sized Value> := <Begin> <Record> <End></p>	<p>Two marks are used to define the boundary of a set or a sequence.</p>

	Since an item in a set or a sequence can itself be a set or a sequence, a recursive definition is used here.
<Selection Criteria> := string	This is the string version of the in parameter (hdselection) for action ftInitiate.
<Interface Name> := string	This is the interface name for the current data object.
< Separator> :=	for example: '\n'
<Begin> :=	for example: "{\n"
<End> :=	for example: "}\n"

Note:

*: 0 or more

+: 1 or more

5.4 Documentation Convention of the Model

The specification of the PM information model is documented in Section 5 and 6. Section 5 covers its requirements, scope, UML model, and the model semantics. Section 6 contains the IDL code for the imported types, forward declarations, structures and typedefs, exceptions, and interfaces. A documentation convention has been used for these sections to improve (1) readability and (2) easy implementation of the specification.

For readability purpose, the descriptive text for the model is included in both Sections 5 and 6. The text in Section 5 is for the people who want to get the idea without reading the IDL. The text (as comments in the IDL code) in Section 6 is for developers to quickly check the model semantics. In addition, subsection numbers and headings are used for the IDL codes, such as imported types, forward declarations, structures and typedefs, exceptions, and the individual interfaces.

For the purpose of easy implementation, subsection numbers and headings of level 2 and lower are encapsulated in IDL comments. In this way, these subsection numbers and headings need not to be removed when extracting the machine processable IDL files from the document.

It should be noted that ThresholdDataFactory and HistoryDataIterator interfaces, though required, are not shown in the table of contents. They are defined in the corresponding section, ThresholdData and HistoryData respectively.

HistoryData object is represented as valuetype rather than interface in this model. However, for the ease of organizing the managed objects and the ease of retrieval, it is listed under "Interfaces" in the "Information Model IDL" section.

6. PM Information Model IDL

```

#ifndef _itut_q822r_idl_
#define _itut_q822r_idl_

#pragma prefix "t1.org"

#include <CosNaming.idl>
#include <itut_x780.idl>
#include <itut_q821.idl>

/**
It is expected that this document will become a delta to ITU Recommendation
Q.822r and the pragma prefix used would be modified from t1.org to itu.int.
This should be taken into consideration when considering future migration and
interoperability.
*/

/**
This module, itut_q822r, contains IDL definition based on objects defined in
X.739 and Q.822. The IDL definitions in this file are the object interfaces.
*/
module itut_q822r
{
/**

```

6.1 Imports

```

IMPORTS
*/

/**
Types imported from itut_x780 and itut_q821
*/
    typedef itut_x780::AdministrativeStateType AdministrativeStateType;
    typedef itut_x780::GeneralizedTimeType GeneralizedTimeType;
    typedef itut_x780::OperationalStateType OperationalStateType;
    typedef itut_x780::PerceivedSeverityType PerceivedSeverityType;
    typedef itut_x780::MO MO;
    typedef itut_x780::MOSetType MOSetType;
    typedef itut_x780::NameType NameType;
    typedef itut_x780::Istring Istring;
    typedef itut_x780::IstringSetType IstringSetType;
    typedef itut_x780::NameBindingType NameBindingType;
    typedef itut_q821::TimeIntervalType TimeIntervalType;

/**
Exceptions imported from itut_x780
*/
#define ApplicationError itut_x780::ApplicationError
#define CreateError itut_x780::CreateError

/**
Interfaces imported from itut_x780
*/
    typedef itut_x780::ManagedObject ManagedObject;
    typedef itut_x780::ManagedObjectFactory ManagedObjectFactory;

/**

```

```
Valuetype imported from itut_x780
*/
#define ManagedObjectValueType itut_x780::ManagedObjectValueType

/**
```

6.2 *Forward Declarations*

```
FORWARD DECLARATIONS
*/

/**
Interface forward declarations
*/
    interface Scanner;
    interface CurrentData;
    interface ZsCurrentData;
    interface ThresholdData;
    interface HistoryDataScanner;
    interface HistoryDataIterator

/**
Valuetype forward declarations
*/
    valuetype ScannerValueType;
    valuetype CurrentDataValueType;
    valuetype ZsCurrentDataValueType;
    valuetype ThresholdDataValueType;
    valuetype HistoryValueType;
    valuetype HistoryDataScannerValueType;
```

/**

6.3 Structures and Typedefs

STRUCTURES AND TYPEDEFS

*/

/**

PerceivedSeverityTypeOpt is an optional type. If the discriminator is true the value is present, otherwise the value is null.

*/

```
union PerceivedSeverityTypeOpt switch (boolean) {
    case TRUE: PerceivedSeverityType    value;
};
```

/**

This data type defines a sequence of HistoryValueType. The order is significant.

*/

```
typedef sequence <HistoryValueType> HistoryValueSeqType;
```

/**

The following definitions are translated from X.739 ASN.1 definitions.

*/

```
enum TimePeriodChoiceType
```

```
{
    daysChoice,
    hoursChoice,
    minutesChoice,
    secondsChoice,
    milliSecondsChoice,
    microSecondsChoice,
    nanoSecondsChoice,
    picoSecondsChoice
};
```

```
union TimePeriodType switch (TimePeriodChoiceType)
```

```
{
    case daysChoice           : unsigned long days;
    case hoursChoice         : unsigned long hours;
    case minutesChoice       : unsigned long minutes;
    case secondsChoice       : unsigned long seconds;
    case milliSecondsChoice  : unsigned long milliSeconds;
    case microSecondsChoice  : unsigned long microSeconds;
    case nanoSecondsChoice   : unsigned long nanoSeconds;
    case picoSecondsChoice   : unsigned long picoSeconds;
};
```

/**

The SeverityIndicatingThresholdType contains the threshold level, which is to be applied to counter/gauge attribute. It shall be initialized when the managed object in which it is included is created and may be modified. An optional parameter is used to associate the threshold level to the severity parameter of the emitted notification. The generation of the notification can be switched off using the boolean parameter notifyOnOff. The severity parameter is mandatory if notifyOnOff is true.

*/

```
struct SeverityIndicatingThresholdType
```

```
{
    float           threshold;
    boolean         notifyOnOff;
};
```

```

        PerceivedSeverityTypeOpt    severityIndication;
};

struct CounterThresholdSettingType
{
    string                attributeId;
    SeverityIndicatingThresholdType    severityIndicatingThreshold;
};

/**
Set type the order is insignificant.
*/
typedef sequence <CounterThresholdSettingType>
                CounterThresholdSetType;

/**
The attribute of SeverityIndicatingGaugeThresholdType has similar
behaviour to the gauge-threshold attribute defined in X.721. The syntax
has an added parameter for indicating the associated severity, as defined
in SeverityIndicatingThresholdType, to the notification triggered by the
crossing of the corresponding threshold level. As an enhancement to the
syntax of the gauge-threshold attribute type it adds an optional severity
indication parameter to the syntax of both the notify-high and notify-low
sub-members within each threshold level member. This attribute type has
additional behavior associated with these optional perceived severity
indication parameters, which is defined as follows:


- If the notify-high's switch is on (true), the notify-high's severity
indication value shall be reported in the perceived severity
parameter of a notification triggered by the gauge value crossing the
notify-high's gauge-threshold value in the positive going direction.
- If the notify-low's switch is on (true), the notify-low's severity
indication value shall be reported in the perceived severity
parameter of a notification triggered by the gauge value crossing the
notify-low's gauge-threshold value in the negative going direction.
- If both switches are on (true) for a single threshold level, one of
the severity indication values shall be "clear". The severity
indicating gauge-threshold shall only emit a clear event notification
if the corresponding threshold level (either notify-high or notify-
low) notification has been emitted and no other clear notification
for this threshold level pair has been emitted since the previous
corresponding threshold level notification has been emitted.


*/
struct SeverityIndicatingGaugeThresholdType
{
    SeverityIndicatingThresholdType notifyLow;
    SeverityIndicatingThresholdType notifyHigh;
};

/**
Seq type the order is insignificant.
*/
typedef sequence <SeverityIndicatingGaugeThresholdType>
                SeverityIndicatingGaugeThresholdSetType;

struct GaugeThresholdSettingType
{
    string attributeId;
    SeverityIndicatingGaugeThresholdSetType
                severityIndicatingGaugeThreshold;
};

/**
Set type the order is insignificant.

```

```
*/
typedef sequence <GaugeThresholdSettingType>
                    GaugeThresholdAttributeSetType;

typedef TimePeriodType TimeOffsetType;

Struct HdSelectionType
{
    NameType          name;
    TimeOffsetType    beginRelativeEndTime;
    TimeOffsetType    endRelativeEndTime;
};

/**
Set type the order is insignificant.
*/
typedef sequence <HdSelectionType> HdSectionSetType;
```

/**

6.4 Exceptions

EXCEPTIONS

*/

/**

6.4.1 Exceptions for Conditional Package

Conditional package exception

*/

```
exception NOperiodSynchronizationPackage {};
```

```
exception NOthresholdPackage {};
```

```
exception NOcounterThresholdListPackage {};
```

```
exception NOgaugeThresholdListPackage {};
```

/**

6.4.2 Exceptions for Performance Management

Editor Note: Placeholder for now.

*/

/**

6.5 Interfaces

INTERFACES

*/

/**

6.5.1 Scanner

A managed object of this interface represents the ability to retrieve values of attributes of managed objects and produce summary information from those values. This summary information may be made available in attributes, notifications, action replies, or some combination of these. Summary information may consist of observed attribute values or statistics calculated from these values (either over time or over managed objects).

Observed attribute values are retrieved during a "scan", which is initiated periodically, at the end of each granularity period, provided that the granularity period is non-zero.

The granularity period attribute indicates the length of the granularity period. The granularity period in the scanner object shall not be modified unless the value of the administrative state is "locked". If the period synchronization package is not present, the time at which a granularity period starts after the scanner is unlocked is a local matter.

The administrative state attribute is used to suspend or resume the scanning function. If administrative state has the value "unlocked", the scanner is administratively permitted to perform scans. If administrative state has the value "locked", the scanner is administratively prohibited from performing scans.

The operational state attribute represents the operational capability of the scanner to perform its functions.

If the attribute value change notification package is present, then changes to the granularity period shall cause attribute value change notifications to be emitted. If the state change notification package is present then changes to the operational state or administrative state shall cause state change notifications to be emitted. If the object create/delete notification package is present, then the creation or deletion of an object of scanner's subclass shall emit the corresponding create or delete notification.

The Scanner interface is not instantiable.

*/

```

valuetype ScannerValueType: ManagedObjectValueType
{
    public AdministrativeStateType      administrativeState;
        // GET-REPLACE
    public TimePeriodType                granularityPeriod;
        // GET-REPLACE
    public OperationalStateType          operationalState;
        // GET
    public GeneralizedTimeType            periodSynchronizationTime;
        // conditional: periodSynchronizationPackage
        // PRESENT IF configurable agent internal synchronization
        // of repeating time periods is required

```

```

// GET-REPLACE

}; // valuetype ScannerValueType

interface Scanner: ManagedObject
{
    /**
    The administrativeState attribute is used to activate and
    deactivate (administratively suspend and resume) the function
    performed by the scanner.

    administrativeState is a read/write attribute supported by Get and
    Set access operations.
    */
    AdministrativeStateType administrativeStateGet()
        raises (ApplicationError);

    void administrativeStateSet(in AdministrativeStateType value)
        raises (ApplicationError);

    /**
    The granularityPeriod attribute is of type TimePeriod that is
    defined in Structures and Typedefs Section. GranularityPeriod
    specifies the length of time during which the "scan" is performed.

    granularityPeriod is a read/write attribute supported by Get and
    Set access operations.
    */
    TimePeriodType granularityPeriodGet()
        raises (ApplicationError);

    Void granularityPeriodSet(in TimePeriodType value)
        raises (ApplicationError);

    /**
    The operationalState attribute identifies whether or not the
    scanning function represented by this object is capable of
    performing its normal functions.

    The attribute operationalState is read-only supported by Get
    access operation.
    */
    OperationalStateType operationalStateGet()
        raises (ApplicationError);

    /**
    Attribute periodSynchronizationTime contains the time to which a
    repeating time period is synchronized. The attribute is of type
    GeneralizedTime. The start of each time period is an integral
    number of periods before or after the time specified by this
    attribute.

    periodSynchronizationTime is a read/write attribute supported by
    Get and Set access operations.
    */
    GeneralizedTimeType periodSynchronizationTimeGet()
        raises (ApplicationError,
            NOperiodSynchronizationPackage);

    void periodSynchronizationTimeSet(in GeneralizedTimeType value)
        raises (ApplicationError,
            NOperiodSynchronizationPackage);
}

```

```
CONDITIONAL_NOTIFICATION
    (ITU_X780::Notifications, objectCreation,
    createDeleteNotificationsPackage);
CONDITIONAL_NOTIFICATION
    (ITU_X780::Notifications, objectDeletion,
    createDeleteNotificationsPackage);
CONDITIONAL_NOTIFICATION
    (ITU_X780::Notifications, attributeValueChange,
    attributeValueChangeNotificationPackage);
CONDITIONAL_NOTIFICATION
    (ITU_X780::Notifications, stateChange,
    stateChangeNotificationPackage);

}; // interface Scanner
```

/**

6.5.2 CurrentData

The currentData object class, a particular type of scanner, is a class of managed support objects that record the current performance data for monitoring purpose. The performance data (measurements) of monitored resources are modeled as attributes in the definition of subclasses of currentData since currentData can't be instantiated. The objects that represent monitored resources contain the corresponding currentData objects in the name binding relationship.

The measurements are collected for a time interval (e.g., 5 min.) specified by granularityTime attribute. At the end of each interval, the elapsedTime attribute will be updated to the difference between the current time and the start of the present monitoring interval. The measurements as well as name, granularityTime, and suspectIntervalFlag are copied into a corresponding history data structure. Further more, the periodEndTime of the history data structure is assigned to the current time.

If historyRetention is greater than zero, then history data structures will be retained in the managed system for at least the duration equivalent to the number of intervals specified in the historyRetention attribute. During that time, history data can be accessed via a set of operations:

- getMostRecent - will retrieve the most recent history data
- getBetween - will retrieve a set of history data within a time window specified by start and end time points

If threshold package is present, the currentData object contains a pointer to a thresholdData object. If any of the thresholds (defined in the referenced thresholdData object) are violated a Quality of Service (QoS) alarm notification is emitted by the currentData object. The thresholdInfo field of the alarm shall contain all the measurement attributes. If additional threshold acrossing(s) occur during the current time interval then additional alarms shall be emitted as well.

New thresholds resulting from modifying the thresholdDataInstance attribute or from changing a threshold value in the referenced thresholdData object, should take effect immediately. If an alarm condition exists previous to the occurrence of a threshold value change (i.e. an old threshold had been violated), and the new threshold value is outside of the range of the old threshold value (e.g., in the case of an increasing counter, the new threshold value is greater than the old threshold value), and the current value of the measurement is within the allowable range of the new threshold value, then a QoS Alarm notification is emitted with a severity of 'clear'. If the new threshold value is set within the range of the old threshold value, such that the new threshold is violated, a QoS Alarm notification is emitted if an alarm condition is not already outstanding.

The thresholdDataInstance attribute contains a set of pointers to thresholdData objects, therefore, more than one QoS Alarm notifications might be emitted for a single monitored attribute. And it is possible the notifications might be conflict. For example, one notification indicates alarm is on, the other cleared. It is the managing systems' responsibility to maintain the notification consistency.

*/

```

    valuetype CurrentDataValueType: ScannerValueType
    {
        public boolean                suspectIntervalFlag;
        // GET REPLACE-WITH-DEFAULT
        public TimeIntervalType        elapsedTime;
    }

```

```

        // GET
public short                                historyRetention;
        // Mandatory now (used to be conditional)
        // GET-REPLACE
public MOSetType                             thresholdDataInstance;
        // conditional: thresholdPackage
        // PRESENT IF a Quality of Service Alarm Notification is
        // to be emitted for threshold crossing
        // GET-REPLACE ADD-REMOVE

}; // valuetype CurrentDataValueType

interface CurrentData: Scanner
{
    /**
    define default value for suspectIntervalFlag
    */
    const boolean suspectIntervalFlagDefault = false;

    /**
    The attribute suspectIntervalFlagGet is used to indicate that the
    performance data for the current period may not be reliable. Some
    reasons for this to occur are:
    

- Suspect data were detected by the actual resource doing data
        collection.
- Transition of the administrativeState attribute to/from the
        'lock' state.
- Transition of the operationalState to/from the 'disabled'
        state.
- The performance counters were reset during the interval.
- The currentData (or subclass) object instance was created
        during the monitoring period.



    This attribute is read-only, supported by Get access operation,
    however it can be replaced by default, which is supported by
    SetDefault operation.
    */
    boolean suspectIntervalFlagGet()
        raises (ApplicationError);

    void suspectIntervalFlagSetDefault()
        raises (ApplicationError);

    /**
    The elapsedTime attribute represents the difference between the
    current time and the start of the present monitoring interval.

    The attribute is read-only, supported by Get access operation.
    */
    TimeIntervalType elapsedTimeGet()
        raises (ApplicationError);

    /**
    The historyRetention attribute specifies the minimum number of
    intervals that the history data structure (just being created)
    must be preserved.

    This attribute is read/write, supported by Get and Set access
    operations.
    */
    short historyRetentionGet()

```

```

        raises (ApplicationError);

void historyRetentionSet(in short value)
    raises (ApplicationError);

/**
The thresholdDataInstance is a "list" attribute in which each item
is a pointer to a thresholdData object that contains threshold
limits for performance parameters. Whenever the value of a PM
parameter violates its threshold setting, a qualityofServiceAlarm
notification is emitted.

This attribute is supported by Get, Set, Add, and Remove access
operations.
*/
MOSetType thresholdDataInstanceGet()
    raises (ApplicationError,
           NOthresholdPackage);

void thresholdDataInstanceSet(in MOSetType value)
    raises (ApplicationError,
           NOthresholdPackage);

void thresholdDataInstanceAdd(in MOSetType value)
    raises (ApplicationError,
           NOthresholdPackage);

void thresholdDataInstanceRemove(in MOSetType value)
    raises (ApplicationError,
           NOthresholdPackage);

/**
Editor Note: The following operations are for the retrieval of
history data.
*/

/**
This method is used to retrieve most recent history data record.
The "most recent" means the most recent history data record saved,
which may be several intervals before the current time because the
historyRetention may be set to zero or the measurements collected
are all zero.

It is possible the getMostRecent may not be able to retrieve any
data back. For example, all measurements are zeros (no history
data is saved) or historyRetention has been set to zero. The
boolean value returned by getMostRecent method indicates the
availability of the data. True means the out parameter contains
the data, false the parameter is undefined.
*/
boolean getMostRecent(out HistoryValueType value)
    raises (ApplicationError);

/**
The getBetween method will retrieve a sequence of history data
records falling within the time window specified by the startTime
and endTime parameters. The resultIterator is used to transfer
large chunk of data.

If howMany is not provided, then the initial amount of data
returned is a local matter.
*/
HistoryValueSeqType getBetween(in GeneralizedTimeType startTime,

```

```
        in GeneralizedTimeType endTime
        in unsigned short howMany
        out HistoryDataIterator resultIterator)
        raises (ApplicationError);

        CONDITIONAL_NOTIFICATION
        (ITU_X780::Notifications, qualityofServiceAlarm,
         NOthresholdPackage);

}; // interface CurrentData
```

/**

6.5.3 ZsCurrentData

ZsCurrentData is a CurrentData with zero suppression. Zero suppression is that when the measurements are all zeros no history data structures will be created. Two attributes are defined for this purpose: numSuppressedIntervals and maxSuppressedIntervals.

```
*/
    valuetype ZsCurrentDataValueType: CurrentDataValueType
    {
        public short                numSuppressedIntervals;
        // GET
        public short                maxSuppressedIntervals;
        // GET-REPLACE
    }; // valuetype ZsCurrentDataValueType

interface ZsCurrentData: CurrentData
{
    /**
    define default value for maxSuppressedIntervals. The default value
    -1 means that no limit on the number of consecutive suppressed
    intervals. On the other hand, the maxSuppressedIntervals is
    effectively equal to infinity.
    */
    const short maxSuppressedIntervals = -1;

    /**
    The numSuppressedIntervals attribute is used to count the number
    of consecutive intervals for which suppression (i.e. non-creation
    of a history data structure) has occurred. This attribute reflects
    performance measurements up to, but not including, the current
    interval. This attribute gets incremented at the end of an
    interval if suppression has occurred. Otherwise, the attribute is
    reset. Or it reaches maxSuppressedIntervals, a history data record
    will be created and it will be reset.

    The attribute is read-only, supported by Get access operations.
    Editor Note: it was read/write in Q.822.
    */
    short numSuppressedIntervalsGet()
        raises (ApplicationError);

    /**
    In conjunction with record compression, the maxSuppressedIntervals
    attribute limits the maximum number of suppressed intervals that
    will be collected without creating a structure of the history data
    (maxSuppressedIntervals = -1 indicates that the maximum number is
    equal to infinity).

    For example, consider an instance of (a subclass of) zsCurrentData
    with maxSuppressedIntervals set to 32, and the interval set to 15
    minutes. For record compression, it means that after 32
    consecutive suppressed (e.g., all-zero) intervals (8 hours) at
    least one history data record (with all zero PM Parameters) will
    be generated with a count of 32. This ensures that at least one
    history data record per maxSuppressedIntervals will be created.
    */
}
```

```
        This attribute is read/write, supported by Get and Set access
        operations.
        */
        short maxSuppressedIntervalsGet()
            raises (ApplicationError);

        void maxSuppressedIntervalsSet(in short value)
            raises (ApplicationError);
}; // interface ZsCurrentData
```

/**

6.5.4 HistoryData (ValueType)

HistoryData object will contain a copy of the performance measurements and other selected attributes that are present in a current data object at the end of the current interval (e.g., 5 min.). A new record of this valuetype structure is created at the end of each interval if historyRetention attribute in the current data object is greater than zero and this record will be retained in the NE for at least the duration equivalent to the number of intervals specified in the historyRetention attribute.

This generic valuetype object models the history data. Specific current data objects (e.g., SDH/SONET current data objects) will be defined with specific history valuetype objects, which are inherited from this generic history valuetype object.

```

*/
    valuetype HistoryValueType
    {
        /**
        Attribute periodEndTime indicates the time at the end of the
        interval.
        */
        public GeneralizedTimeType        periodEndTime;
            // GET

        /**
        Attribute granularityPeriod is used to copy the same attribute
        from the currentData object.
        */
        public TimePeriodType              granularityPeriod;
            // GET

        /**
        Attribute suspectIntervalFlag is used to copy the same attribute
        from the currentData object.
        */
        public boolean                      suspectIntervalFlag;
            // GET

        /**
        Attribute numSupressedIntevals is used to copy the same attribute
        from the zsCurrentData object. When copying from currentData
        object, it uses 0 as default value.
        */
        public numSupressedIntevalsType    numSupressedIntevals;
            // GET

    }; // valuetype HistoryValueType

    /**
    HistoryDataIterator is used for iteratively transferring large chunk of
    history data.
    */
    interface HistoryDataIterator
    {
        /**
        This method is used to return the next how many history data

        param howMany                                Maximum number of history data
                                                    for which results should be
                                                    returned in first batch, must
                                                    be non-zero

```

```
param HistoryDataSeqType      Sequence of history data
param boolean                 true if there is something in
                               the out param, false otherwise
*/
boolean getNext (in unsigned short howMany out HistoryValueSeqType)
    raises (ApplicationError);

/**
This method is used to destroy the iterator and release its resources.
This must be done by the application.
*/

void destroy ();

}; // interface HistoryDataIterator
```

/**

6.5.5 ThresholdData

The thresholdData object class is a class of managed support objects that contains the values of the threshold settings for the PM parameters. At least one of the counterThresholdListPackage or the gaugeThresholdListPackage must be instantiated.

Thresholds are established by use of the managed object class thresholdData. The thresholdData objects specify the thresholds being applied. Threshold values in a thresholdData object may apply to multiple currentData objects and thresholdData objects are pointed to by currentData objects. The currentData, or its subclasses, object indicates the collection interval used with the threshold. A qualityOfServiceAlarm notification is generated whenever a threshold is crossed.

A threshold may be specific to a single managed object. In this case, the thresholdData object is contained within the managed object being observed. A currentData object within the managed object points to a thresholdData object. This linkage is needed for consistency with the multiple managed object case.

It is sometimes desirable to have a threshold apply to a group of managed objects. In this case, the thresholdData object is external to the managed object being observed. It may be contained within the managedElement object. The thresholdData object is pointed to by all the currentData objects to which it applies.

If createDeleteNotificationsPackage is present, the creation or deletion of a threshold object will cause a objectCreation or objectDeletion notification be emitted.

If attributeValueChangeNotificationPackage is present, then any threshold change will cause an attributeValueChange notification being emitted.

```
*/
    valuetype ThresholdDataValueType: ManagedObjectValueType
    {
        public CounterThresholdSetType
            counterThresholdList;
        // conditional: counterThresholdListPackage
        // PRESENT IF an instance supports it and the
        // gaugeThresholdListPackage is not present
        // GET-REPLACE ADD-REMOVE

        public GaugeThresholdAttributeSetType
            gaugeThresholdList;
        // conditional: gaugeThresholdListPackage
        // PRESENT IF an instance supports it and the
        // counterThresholdListPackage is not present
        // GET-REPLACE ADD-REMOVE

    }; // valuetype ThresholdDataValueType

    interface ThresholdData: ManagedObject
    {
        /**
        The counterThresholdList 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 attribute is supported by Get, Set, Add, and Remove access operations.

```

*/
CounterThresholdSetType
    counterThresholdListGet()
    raises (ApplicationError,
           NOcounterThresholdListPackage);

void counterThresholdListSet(in
    CounterThresholdSetType value)
    raises (ApplicationError,
           NOcounterThresholdListPackage);

void counterThresholdListAdd(in
    CounterThresholdSetType value)
    raises (ApplicationError,
           NOcounterThresholdListPackage);

void counterThresholdListRemove(in
    CounterThresholdSetType value)
    raises (ApplicationError,
           NOcounterThresholdListPackage);

/**
The gaugeThresholdList attribute contains a set of threshold
settings for performance attributes of the gauge type. Each
threshold setting consists of the attribute identifier, the
threshold value and (optionally) the severity of the threshold-
exceeded event.

The attribute is supported by Get, Set, Add, and Remove access
operations.
*/
GaugeThresholdAttributeSetType
    gaugeThresholdListGet()
    raises (ApplicationError,
           NOgaugeThresholdListPackage);

void gaugeThresholdListSet(in
    GaugeThresholdAttributeSetType value)
    raises (ApplicationError,
           NOgaugeThresholdListPackage);

void gaugeThresholdListAdd(in
    GaugeThresholdAttributeSetType value)
    raises (ApplicationError,
           NOgaugeThresholdListPackage);

void gaugeThresholdListRemove(in
    GaugeThresholdAttributeSetType value)
    raises (ApplicationError,
           NOgaugeThresholdListPackage);

CONDITIONAL_NOTIFICATION
    (ITU_X780::Notifications, objectCreation,
     createDeleteNotificationsPackage);
CONDITIONAL_NOTIFICATION
    (ITU_X780::Notifications, objectDeletion,
     createDeleteNotificationsPackage);
CONDITIONAL_NOTIFICATION
    (ITU_X780::Notifications, attributeValueChange,
     attributeValueChangeNotificationPackage);

```

```
}; // interface ThresholdData

interface ThresholdDataFactory: ManagedObjectFactory
{
    ThresholdData create
        (in NameBindingType nameBinding,
         in MO superior,
         inout Istring name, // auto naming if null
         in IstringSetType packageNameList,

         // parameters from ThresholdData
         //
         in CounterThresholdSetType
             counterThresholdList,
             // conditional: counterThresholdListPackage
             // GET-REPLACE ADD-REMOVE
         in GaugeThresholdAttributeSetType
             gaugeThresholdAttributeList
             // conditional: gaugeThresholdListPackage
             // GET-REPLACE ADD-REMOVE
         )
        raises (ApplicationError, CreateError);
}; // interface ThresholdDataFactory
```

/**

6.5.6 HistoryDataScanner

HistoryDataScanner is designed to retrieve history data saved under the relevant currentData objects.

There are some open issues (listed in Section 7) need to be resolved before the design can be completed.

```

*/
    valuetype HistoryDataScannerValueType: ScannerValueType
    {
        public HdSelectionSetType      hdSectionList;
        // GET-REPLACE

    }; // valuetype HistoryDataScannerValueType

interface HistoryDataScanner: Scanner
{
    /**
    The hdSectionList attribute is used to specify which history data
    to be retrieved by history data scanner.

    This attribute is supported by Get, Set, Add, and Remove access
    operations.
    */
    HdSelectionSetType hdSectionListGet()
        raises (ApplicationError);

    void hdSectionListSet(in HdSelectionSetType value)
        raises (ApplicationError);

    void hdSectionListAdd(in HdSelectionSetType value)
        raises (ApplicationError);

    void hdSectionListRemove(in HdSelectionSetType value)
        raises (ApplicationError);

    /**
    Editor Note: more capabilities will be added when the open issues
    get resolved.
    */
}; // interface HistoryDataScanner

```

/**

6.6 Notifications

Editor Note: Placeholder for now.

*/

```
interface Notifications {  
  
}; // interface Notifications
```

/**

6.7 Name Binding

NAME BINDING

*/

/**

The following module contains name-binding information.

*/

```

    module NameBinding
    {

```

/**

6.7.1 ThresholdData

*/

/**

This name binding is used to name the ThresholdData object to a Managed Element object.

*/

```

module ThresholdData_ManagedElement
{

```

{

```

    const string superiorClass =
        "itut_m3120::ManagedElement";
    const boolean superiorSubclassesAllowed = TRUE;
    const string subordinateClass =
        "itut_q822r::ThresholdData";
    const boolean subordinateSubclassesAllowed = TRUE;
    const boolean managerCreatesAllowed = TRUE;
    const DeletePolicyType deletePolicy =
        itut_x780::deleteOnlyIfNoContainedObjects;
    const string kind =
        "itut_q822r::ThresholdData";

```

}; // module ThresholdData_ManagedElement

}; // module NameBinding

}; // module itut_q822r

#endif // _itut_q822r_idl_

7. Open Issues

-
- How is the historyData scan report encoded?
-
-
-

ITU - Telecommunication Standardization Sector
UIT - Sector de Normalización de las Telecomunicaciones

Study Period 2001-2004

Commission d'études ;Study Group;Comisión de Estudio} 4

Contribution tardive;Delayed Contribution ;Contribución tardía} **D.xxx**

Geneva, 15 - 19 January, 2001

Texte disponible seulement en ;Text available only in;Texto disponible solamente en} **E**
Question(s): Q14/4, 15/4, 18/4, 19/4 (old numbers)

SOURCE*: T1M1 (Proposed United States of America)

TITLE: CORBA Alarm Management Model

This contribution proposes a new recommendation to supplement Recommendation Q.821 to supply a Recommendation Q.821-based Management Information Model to be used in telecommunications network management based on CORBA. It defines the model using Interface Definition Language (IDL). The intent of this document is to define a CORBA / IDL model similar to that defined in Recommendation Q.821 using CMISE. This document is compliant with proposed CORBA modeling standards Recommendation X.780, Recommendation Q.816 and Recommendation M.3120.

* **Contact:** Randall J. Scheer
Lucent Technologies
Room 1E225
6200 East Broad Street
Columbus, OH 43213 USA
Tel: +1 614-860-4530
Fax: +1 614-868-3849
E-mail: rjscheer@lucent.com

Attention: This is not a publication made available to the public, but **an internal ITU-T Document** intended only for use by the Member States of the ITU, by ITU-T Sector Members and Associates, and their respective staff and collaborators in their ITU related work. It shall not be made available to, and used by, any other persons or entities without the prior written consent of the ITU-T.

1. INTRODUCTION

This contribution proposes a new recommendation to supplement Recommendation Q.821 [7] to supply a Recommendation Q.821-based Management Information Model to be used in telecommunications network management based on CORBA. It defines the model using Interface Definition Language (IDL). The intent of this document is to define a CORBA / IDL model similar to that defined in Recommendation Q.821 using CMISE. This document is compliant with proposed CORBA modeling standards Recommendation X.780 [16], Recommendation Q.816 [6] and Recommendation M.3120 [3].

This document has the following sections:

Section 1	Introduction
Section 2	References
Section 3	Summary of proposed Recommendation Q.821 changes
Section 4	Scope and purpose
Section 5	Document conventions
Section 6	Alarm surveillance
Section 7	Alarm synchronization
Section 8	Alarm synchronization relationship with other documents
Section 9	Conformance
Section 10	Recommendation Q.821 IDL listing
Section 11	Potential changes

2. REFERENCES

- [1]. ITU-T, *M.3010 – Principles For A Telecommunications Management Network*, draft, February, 2000.
- [2]. ITU-T, *M.3100 – Generic Network Information Model – Amendment 3*, draft
- [3]. ITU-T, *M.3120 – CORBA Generic Network And Network Element Level Information Model*, draft, August, 2000.
- [4]. ITU-T, *M.3400 – TMN Management Functions*, draft, February, 2000.
- [5]. ITU-T, *Q.68 – Overview Of Methodology For Developing Management Services*, March, 1993.
- [6]. ITU-T, *Q.816 – CORBA Bases TMN Services*, draft, August, 2000.
- [7]. ITU-T, *Q.821 – Stage 2 And Stage 3 Description For The Q3 Interface – Alarm Surveillance*, Draft, February, 2000.
- [8]. ITU-T, *X.701 – Systems Management Overview*, August, 1997.
- [9]. ITU-T, *X.710 – Common Management Information Service*, October, 1997.
- [10]. ITU-T, *X.722 – Guidelines For The Definition Of Managed Objects*, 1992.
- [11]. ITU-T, *X.731 – State Management Function*, January, 1992.
- [12]. ITU-T, *X.733 – Alarm Reporting Function*, 1992.
- [13]. ITU-T, *X.734 – Event Report Management Function*, 1993.
- [14]. ITU-T, *X.734 – Event Report Management Function – Amendment 3*, draft.
- [15]. ITU-T, *X.735 – Log Control Function*, September, 1992.
- [16]. ITU-T, *X.780 - TMN Guidelines for Defining CORBA Managed Objects*, draft, August, 2000.
- [17]. ITU-T, *X.792 - Configuration Audit Support Function For ITU-T Applications*, March, 1999.
- [18]. OMG, *Event Service Specification*, June, 2000.

- [19]. OMG, *Notification Service*, June, 2000.
- [20]. OMG, *Telecom Log Service Specification*, Version 1.0, January, 2000.
- [21]. OMG, *Trading Object Service Specification*, June, 2000.
- [22]. T1M1.5, *Working Document For Draft Standard ANSI T1.2xx-2000, CORBA Generic Network And NE Level Information Model*, October, 2000.
- [23]. T1M1.5, *Working Document For Draft Standard ANSI T1.2xx-2000, Framework For CORBA-Based Telecommunications Management Network Interfaces*, October, 2000.

3. PROPOSAL

This contribution proposes a new recommendation to supplement Recommendation Q.821 to supply a Recommendation Q.821-based Management Information Model to be used in telecommunications network management based on CORBA. The intent of this document is to provide the functions and services definitions for CORBA-based interface (with supplied IDL (see section 10)). The intent is not to change the current Q / CMIP interfaces defined in Recommendation Q.821 [7].

In this document, references to Recommendations X.780 [16] and Q.816 [6] are made instead of referencing reference [23]. Reference [23] is the T1M1.5 version of Recommendations X.780 and Q.816. Similarly, this document references to Recommendation M.3120 [3] are made instead of referencing reference [22]. Reference [22] is the T1M1.5 version of Recommendation M.3120.

The rest of this section summarizes the changes from Recommendation Q.821.

3.1 Use Of OMG Notification Service Instead Of Event Reporting

The OMG Notification Service has a different interface from Q / CMISE-based event notifications.

Q / CMISE-based event notifications use a manager / agent model [13]. The manager has the agent create one or more Event Forwarding Discriminators (EFDs) for them. The EFDs determine which notifications will be sent to the manager. When the agent emits an event notification, it checks each of its EFD filters to determine if that manager should be sent that particular event notification.

The OMG Notification Service [19] uses a client / server model. The agent is a supplier of event notifications and the manager is a consumer of event notifications. Both the agent and manager are clients to the OMG Notification Service server. Both the agent and manager can define filters to see which event notifications are maintained by the OMG Notification Service and which event notifications are distributed to different consumers. In this model, the agent does not need to know which managers are consuming their event notifications.

Figure 3-1 gives a graphical representation of the OMG Notification Service.

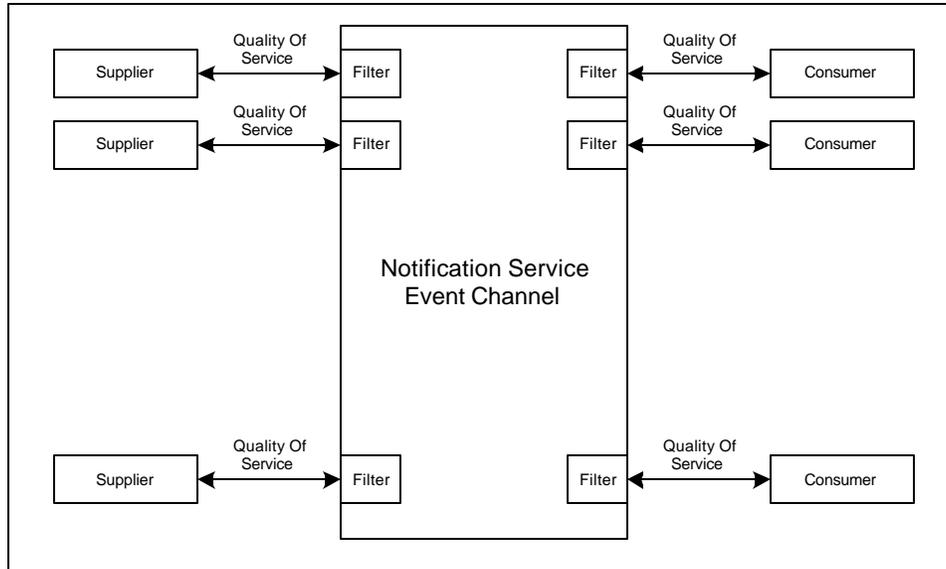


Figure 3-1. OMG Notification Service

The OMG Notification Service uses a different grammar for defining filters. An EFD uses a CMIS Filter construct for defining filters [13]. Recommendation Q.816 [6] uses an enhanced filtering construct based from the OMG Notification Service Extended TCL grammar (which is based from the OMG Trader Constraint Language (TCL) [21]).

Q / CMISE-based event notifications provide the capability to respond to notifications. The CORBA Framework does not offer this capability.

In this model, Alarm Synchronization will use the enhanced filtering grammar defined in Recommendation Q.816 [6]. This is an enhancement to the OMG Notification Service Extended TCL grammar.

Figure 3-2 shows a subset of the OMG Notification Service objects. Other objects and combinations are possible (including, pull consumers, pull suppliers, typed push consumers, OMG Event Service objects [18], etc.).

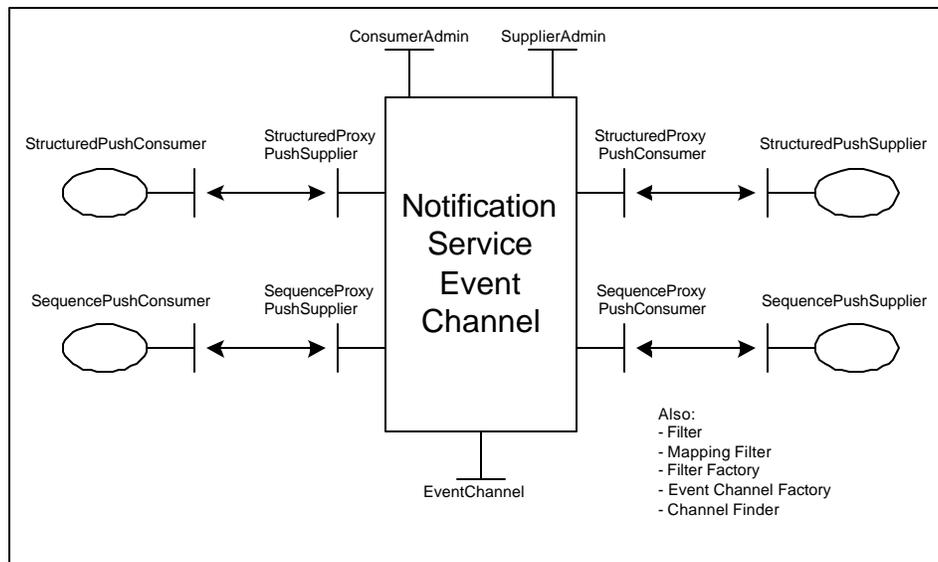


Figure 3-2. Subset Of OMG Notification Service Objects

Recommendation Q.816 supports the following types of notification consumers and suppliers:

- 1) Push Structured Event.
- 2) Push Event Batches (i.e., set of Structured Event).
- 3) Push Typed Events.
- 4) Pull Structured Event.
- 5) Pull Event Batches.

The pull of typed events is not currently supported. Note that consumers and suppliers do not need to be of the same type; OMG Notification Service performs a conversion between the different types of communication. (There is an exception: suppliers who supply Structured Event and Event Batch notifications cannot supply to consumers who want Typed Events.)

Recommendation Q.816 requires that all OMG Notification Service Event Channels be registered with the Channel Finder interface.

3.2 Use Of OMG Telecom Log Service Instead Of Log

As defined in Recommendation Q.816 [6], the OMG Telecom Log Service [20] is used instead of Recommendation X.735 [15] Logs and Log Records. Figure 3-3 gives a graphical representation of the OMG Telecom Log Service log.

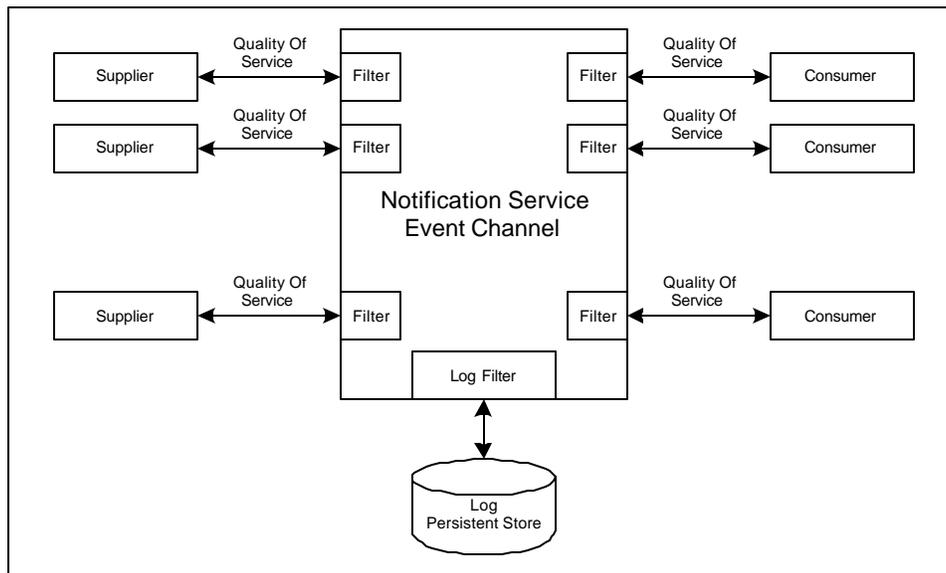


Figure 3-3. OMG Telecom Log Service

With the OMG Telecom Log Service, logs are implemented as OMG Notification Service [19] Event Channels. Notifications supplied to the log are stored as Log Records. Notifications supplied to a log may also be forwarded to other logs or to other applications. Logs may also generate their own notifications (as an example, log is full). Managers may create their own logs and supply a notification filter to filter which notifications are logged. The OMG Telecom Log Service uses the OMG Notification Service Extended TCL grammar.

In the OMG Telecom Log Service model, logs may be created and changed by managers without the direct involvement of agent systems.

In the OMG Telecom Log Service, log records are not sub-classed by their type (such as Alarm Record versus State Change Record).

Notifications of type Structured Event, Event Batch or Typed Event may be stored in a Notify Log. Notifications of type Typed Event may be stored in a Typed Notify Log.

Since the OMG Telecom Log Service is embedded as part of the OMG Notification Service, notifications are issued before log records are created. This implies that Log Record Ids are not known before notifications are issued. Thus, the Correlated Record Name, Correlated Record Name Action, Log Record Id and Log Record Id Action parameters dealing with Log Record Id have been deleted.

Recommendation Q.816 requires that all OMG Telecom Log Service Event Channels be registered with the Channel Finder interface.

3.3 Different top Managed Object

In Recommendation X.780 [16], it is defined that all managed objects are sub-classed from the Managed Object object, instead of the top managed object. Refer to Recommendation X.780 for a description of the Managed Object object.

3.4 Remove Naming Attributes

In Recommendation X.780 [16], attributes only used for naming are not required with a CORBA managed object interface.

This model removes the GDMO naming attributes with each managed object.

3.5 Remove Cancel Alarm Synchronization Action

The concept of Cancel Alarm Synchronization is no longer required. First, with CORBA, you can't cancel a method that is currently under progress. Secondly, the Alarm Synchronization action has been changed to include an iterator that can be destroyed (cancelled) once Alarm Synchronization has partially returned. In this document, destroying the Alarm Synchronization iterator is analogous to the Q / CMISE Cancel Alarm Synchronization action.

This model removes the Cancel Alarm Synchronization action and the No Such Invoke Id Error Parameter and Canceled Alarm Synchronization Parameter parameters

3.6 No Longer Require Parameter Re-Definition

In Recommendation Q.821 [7], when performing the Alarm Synchronization action, EVENT-INFO parameters used in alarm event notifications needed to be redefined as ACTION-REPLY parameters. This is because the Alarm Synchronization action needed to return the parameter information as part of the Additional Information attribute and only ACTION-REPLY parameters are acceptable in actions. In Q / CMIP, EVENT-INFO parameters can only be used in event notifications and ACTION-REPLY parameters can only be used in action responses [10].

In the CORBA Framework [16], parameters are not designated as ACTION-REPLY or EVENT-INFO parameters. This means that the same parameter that is used to supply information into the Additional Information attribute in an event notification can be used as part of the Alarm Synchronization method. Parameters do not need to be re-defined.

This document removes the Suspect Object List Action Parameter.

3.7 Suspect Object List Moved To Recommendation X.780

The Suspect Object List parameter is optionally used in alarms to show a probability of which managed objects possibly caused the alarm. It is defined in Recommendation Q.821 [7], but not used in Recommendation Q.821. The definition for this parameter has been moved to Recommendation X.780 [16], where it is optionally included in alarms.

3.8 Changes To Alarm Info

In Recommendation X.780 [16], a number of changes have been made to the CMIP Alarm Info container. It is important that Alarm Synchronization uses the same containers as used with OMG Notification Service [19]. The container used for sets of notifications in the OMG Notification Service is Event Batch. Note that Typed Events may be mapped so that they can be contained in an Event Batch (see section 7.2.5.1.2).

The types for each of the Alarm Info parameters has changed (as an example, how a Managed Object Instance is represented is different). However, parameters of the same type have each been changed in a consistent way (as an example, all parameters using Managed Object Instances use them in the same way).

The Event Time and Notification Identifier parameters are now mandatory.

The following new parameters have been added to Alarm Info:

1. Alarm Effect On Service – Optional True or False indication whether the alarm has an effect on service.
2. Alarming Resumed – Optional True or False indication if alarming was just resumed, possibly resulting in the delayed reporting of an alarm. Used as part of Alarm Reporting Control [2].
3. Suspect Object List – Optional list of managed objects (potentially with probability) that may have caused the alarm.

3.9 Action Results Without The Use Of Linked Replies

In Q/CMISE [9], multiple replies to a single management operation may occur with an action operation for a single managed object instance in which the action is defined to produce multiple results. These multiple replies will be linked and may contain multiple success and failure replies.

In the CORBA framework [16], all actions are returned in a single invocation. If multiple invocations are required for some reason (such as the results may be too large), an iterator must be used. Each invocation can throw only a single exception (of course, any exception may include multiple data items).

In this model, Alarm Synchronization could have results that are larger than can be returned by a single CORBA method, so an iterator managed object is used to allow multiple invocations.

3.10 Use Of Channels Instead Of EFD Associations

In Q/CMISE [13], Event Forwarding Discriminators (EFDs) contain associations for how to send notifications from an agent to a manager. In Recommendation Q.816 [6], the OMG Notification Service [19] is used to send notifications from an agent to a manager. As a result, the Route Current Alarm Summary and Request Current Alarm Summary Route functions are no longer required.

The Management Operations Schedule, from Recommendation Q.821 [7], contains the Destination Address attribute. It is used to allow the sending of the Current Alarm Summary Report notification to different associations. In the CORBA Framework, what Event Channels are used to send notifications are outside the scope of an object. Thus, a managed object cannot choose to send a message via one Event Channel or another. In this model, the Destination Address has been removed from the Management Operations Schedule object.

3.11 Use Of CORBA Naming Conventions

Recommendation X.780 [16] has naming conventions for defining attribute names.

In this model, Recommendation X.780 naming conventions will be used. As an example, the new name for ObjectList is ObjectListSetType.

3.12 ASN.1 Definitions Not Used In Recommendation Q.821

Recommendation Q.821 [7] defines a few ASN.1 definitions that are not directly used or referenced in Recommendation Q.821.

In this model, only the ASN.1 definitions that are either directly used by Recommendation Q.821 or discovered to be used by other Recommendation standards have been defined. The results are that NotificationId, ProblemData, StatusChange, CountInterval, CountWindow, ValueDuration, GaugeParameters and Threshold are not defined.

3.13 Changes To Audible Visual Local Alarm Handling

Recommendation M.3120 [3] changes the allowAudibleVisualLocalAlarm and inhibitAudibleVisualLocalAlarm actions into an attribute. The actions are performed by setting the attribute. This allows the value of the attribute to be retrieved via a Get operation.

3.14 Heartbeat Service

Recommendation Q.816 [6] defines a new service called “Heartbeat Service” that allows the sending and control over a periodic heartbeat notification.

3.15 Complex Constants

As stated in recommendation X.780 [16], CORBA IDL only allows constants to be defined for base types and enumerated types. If an attribute type is complex, no default can be defined for it.

In this model, the default scope cannot be defined because it is of a complex type.

3.16 CMISE Services Different From CORBA Services

Recommendation Q.821 [7] specifies each of the Fault Management functions and services. Where the functions are generic and can, in general, be maintained when using CORBA, the Recommendation Q.821 services are Q / CMISE specific. As an example, the Q / CMISE services specifically mention the Event Forwarding Discriminator managed object, which is not provided when using CORBA.

This document specifies the services needed with the CORBA Recommendation Q.821 solution (see section D.1).

The following functions have been added for the CORBA solution set:

- Request Inhibit/Allow Audible and Visual Local Alarms Indications

The following services have been added for the CORBA solution set:

- Get Event Channel
- Get Heartbeat Period
- Get Inhibit/Allow Audible and Visual Local Alarms
- Heartbeat
- Initiate Event Channel
- Log Lookup
- Obtain Event Channels
- Set Event Channel
- Set Heartbeat Period
- Set Inhibit/Allow Audible and Visual Local Alarms
- Terminate Event Channel

The following functions have been deleted when moving from the Q / CMISE solution set to the CORBA solution set:

- Request Current Alarm Summary Route
- Route Current Alarm Summary

The following services have been deleted when moving from the Q / CMISE solution set to the CORBA solution set:

- Get Event Forwarding Discriminator
- Inhibit/Allow Audible and Visual Local Alarms
- Set Event Forwarding Discriminator

4. SCOPE AND PURPOSE

4.1 Scope

This Recommendation is part of a series of Recommendations that specify the CORBA interface requirements for communication between an Operations System (OS) and a Network Element (NE), between an OS and a Mediation Device (MD), between an OS and a Q Adapter (QA), and between OSs in a Telecommunication Management Network (TMN) [1]. The current issue of this Recommendation provides a Stage 2 and Stage 3 Description [5] for Alarm Surveillance to support the associated TMN management service component described in Recommendation M.3400 [4].

4.2 Purpose

Current telecommunications networks are populated by a large and increasing number of OSs and network elements supplied by different vendors. Both the number and variety of networks and services have grown, creating a diversity of management needs. This growth has resulted in the proliferation of unique communication interfaces between OSs and network elements. The telecommunications industry stands to benefit from the standardization of these interfaces, designed to achieve interoperability between a broad range of OSs and network elements / Q Adapters, using Mediation Devices where appropriate, and between OSs.

The primary purpose of this Recommendation is to provide a set of application messages and associated support objects for the support of communication across CORBA interfaces. Because of the desirability of providing common TMN solutions, these messages and support objects are expected to be applicable to other TMN or TMN-related interfaces.

4.3 Definitions From Other Documents

Agent	[1]
Alarm	[12]
Alarm Event	[7]
Alarm Info	[16]
Alarm Reporting	[12]
Alarm Status	[7]
Alarm Surveillance	[7]
Alarm Synchronization	[7]
Correlated Notifications	[12]
Current Alarm	[7]
Element Management Layer	[1]
Fault Management	[4]
Inheritance Hierarchy	[16]
Management Information Model	[1]
Manager	[1]
Naming Tree	[16]
Network Management Layer	[1]
Notification Identifier	[12]
Subordinate Objects	[16]
Superior Object	[16]
System Management Functional Unit	[8]

4.4 Abbreviations

For the purpose of this Recommendation, the following abbreviations are used:

ASN.1	Abstract Syntax Notation One
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Element

Cnf	Confirm
CORBA	Common Object Request Broker Architecture
EFD	Event Forwarding Discriminator
GDMO	Guidelines For The Definition Of Managed Objects
IDL	Interface Definition Language
Ind	Indication
MAPDU	Management Application Protocol Data Unit
MD	Mediation Device
MIB	Management Information Base
MOO	Multiple Object Operation
NE	Network Element
OMG	Object Management Group
OS	Operations System
QA	Q Adapter
QOS	Quality Of Service
RDN	Relative Distinguished Name
Req	Request
Rsp	Response
TCL	Trader Constraint Language
TMN	Telecommunications Management Network

5. CONVENTIONS

The definition of several Alarm Surveillance services in this Recommendation includes a table that lists the parameters of its primitives. For a given primitive, the presence of each parameter is described by one of the following values:

M	The parameter is mandatory
(=)	The value of the parameter is equal to the value of the parameter in the column to the left
U	Use of the parameter is a service-user option
O	Optional. Optionality is subject to definition according to the Service Level Agreement or Contract between the manager and agent, i.e. a parameter listed as optional may be made mandatory by the Contract
–	The parameter is not present in the interaction
C	The parameter is conditionally present. The condition(s) are defined by the text that describes the parameter
P	Subject to the constraints imposed on the parameter by Recommendation X.780 [16]

Except for OS - OS communications, the term managing system refers to the OS and the term managed system refers to a network element, Q Adapter or a Mediation Device. Network elements may be exchanges, signalling systems or other network resources as specified in other Recommendations that reference this Recommendation. For OS - OS communications, one OS is the managing system while the other is the managed system.

6. ALARM SURVEILLANCE MANAGEMENT INFORMATION

This section describes the semantics of management information related to Alarm Surveillance.

6.1 Managed Object Classes

The Alarm Surveillance Services specified below are applicable to the managed object classes of an information model specified in any other Recommendation if the proper references to this Recommendation are made in the relevant managed object classes. In particular, these services are applicable to the managed object classes of the Generic Network Information Model [3].

6.2 Support Object Classes

The following support object classes (or their subclasses) [3], support the Alarm Surveillance functions specified in this Recommendation:

- Alarm Severity Assignment Profile
- Managed Element

The containment relationships between these support object classes are shown in Figure 6-1 using the Entity-Relationship notation as in [3].

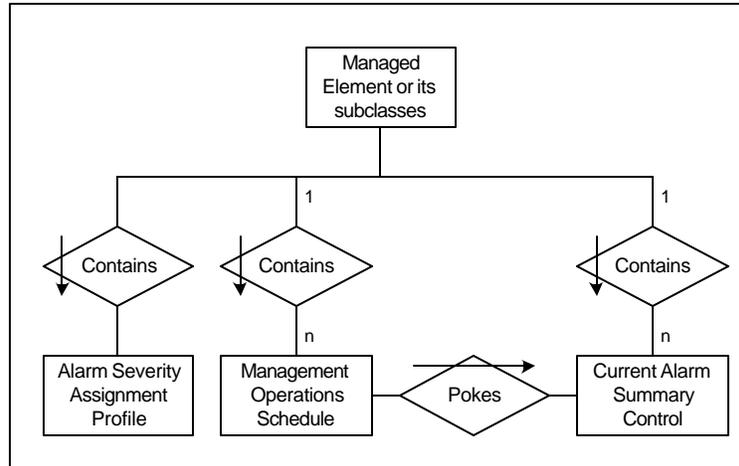


Figure 6-1. Containment Relationship Between Alarm Surveillance Support Objects

6.2.1 Current Alarm Summary Control

The Current Alarm Summary Control object class is a class of support objects that provide the criteria for generation of current alarm summary reports. An object is included in a current alarm summary report if:

- The object is included in the Object List, (if the list is non-empty),
- The object has an Alarm Status that is present in the Alarm Status List (if the list is non-empty) and
- The object has an alarm (or potential alarm) with a Perceived Severity and Probable Cause matching members of the Perceived Severity List (if non-empty) and Probable Cause List (if non-empty), respectively.

If the Object List is empty then the criteria in the Current Alarm Summary Control are applied to all objects in the Managed System. If any of the other criteria are empty then they are not used in selecting objects that will appear in the current alarm summary report.

A single object may appear in a report multiple times if it has multiple outstanding alarm conditions that match the Perceived Severity List and Probable Cause List criteria.

This object class is a subclass of the Managed Object object class.

The semantics of associated attributes are as follows:

a) *Alarm Status List*

The Alarm Status List attribute type describes criteria for inclusion in a current alarm summary report. The Alarm Status List consists of a set of possible Alarm Status values. In order to be included in a current alarm summary report, an object shall have an Alarm Status that matches one of the states in the Alarm Status List.

If the Alarm Status List has null value, the Alarm Status of the objects in the Object List is not used as a criterion for inclusion in the current alarm summary report.

b) *Object List*

The Object List attribute type describes a set of object instances.

c) *Perceived Severity List*

The Perceived Severity List attribute type describes criteria for inclusion in a current alarm summary report. It consists of a set of possible Perceived Severity values. In order to be included in a current alarm summary report, an object must have an outstanding alarm (or potential alarm) that has a Perceived Severity that matches one of the elements in the Perceived Severity List.

If the Perceived Severity List has null value, the Perceived Severity of the objects in the object list is not used as a criterion for inclusion in the current alarm summary report.

d) *Probable Cause List*

The Probable Cause List attribute type describes criteria for inclusion in a current alarm summary report, consisting of a set of possible Probable Cause values. In order to be included in a current alarm summary report, an object must have an outstanding alarm (or potential alarm) that has a Probable Cause that matches one of the elements in the Probable Cause List.

If the Probable Cause List has a null value, the Probable Cause of the objects in the object list is not used as a criterion for inclusion in the current alarm summary report.

6.2.2 Management Operations Schedule

The Management Operations Schedule object class is a class of support objects that provide the ability to schedule a management service to occur periodically. The period is specified by an Interval, with the first occurrence of the service (coinciding with the start of the first interval) specified as the Begin Time. The end of the time span during which the service can occur is defined by the End Time.

The object(s) that will supply the service are defined by the Affected Object Class and Affected Object Instances (e.g., the Current Alarm Summary Control object when providing the Current Alarm Summary Reporting service). The Administrative State is used to allow/inhibit the operation of the schedule. The Operational State optionally describes whether the object is capable of performing its functions.

This object class is a subclass of the Managed Object object class.

The semantics of associated attributes are as follows:

a) *Administrative State*

The semantics of the Administrative State attribute type are described in Recommendation X.731 [11]. It can be used to suspend and resume the Management Operations Schedule.

b) *Affected Object Class*

The Affected Object Class attribute type identifies the object class affected by a scheduled management operation.

c) *Affected Object Instances*

The Affected Object Instances attribute type identifies the object instances on which a scheduled management operation will be performed.

d) *Begin Time*

The Begin Time attribute type indicates the starting time for a management function.

e) *End Time*

The End Time attribute type indicates the termination time of a management function.

f) *Interval*

The Interval attribute type indicates the time between occurrences of a given activity described by an instance of the Management Operations Schedule object class. The interval can be specified in seconds, minutes, hours or days.

g) *Operational State*

The semantics of the optional Operational State attribute type are described in Recommendation X.731.

7. ALARM SYNCHRONIZATION

7.1 Overview Of Alarm Synchronization

This section gives a brief summary of the Alarm Synchronization capabilities.

7.1.1 Introduction

Many Fault Management systems need to maintain a collection of uncleared alarm conditions. This allows systems providing alarm surveillance to provide their users a description of current network faults. Following communication outages and other system failures, this collection needs to be synchronized with the agent's complete set of uncleared alarm conditions. (We are calling this collection of uncleared alarms as "Current Alarms", see section 7.1.2.) In addition, many manager systems need the full complement of alarm information.

Managers may need to synchronize their alarm databases during the following situations:

- 1) A communications loss of either short or long duration.
- 2) Serious problems within the manager (e.g., a disk crash).
- 3) An operator error (e.g., inadvertent deletion of alarms).
- 4) The initial manager to agent connection.
- 5) Verification of whether alarm conditions are still pending.
- 6) Modification of an OMG Notification Service filter [19].

7.1.2 Current Alarms

Current Alarms are active alarms that have not yet been cleared. Alarms become current when they are initially emitted as notifications. Alarms are no longer current when they are cleared by notifications. Alarms may also no longer be current when their managed object instance has been deleted. The process of clearing alarms is further discussed in Appendix A.

In this standard, it is the agent's responsibility to maintain the collection of Current Alarms. Current Alarms are not required to be maintained in managed objects.

7.1.3 Itemized Alarm Synchronization Requirements

This section describes the Alarm Synchronization requirements. The Alarm Synchronization requirements have been updated so that they do not use Q / CMISE-specific terms. The following requirements can be used for both the Q / CMISE and CORBA solutions.

The Alarm Synchronization service requirements are as follows:

- 1) Put in line the manager with the information contained in the agent about Current Alarms (i.e. alarms not yet cleared) at the time of the Alarm Synchronization request.
- 2) Alarm Synchronization and Alarm Reporting will operate independently (in both manager and agent) from the interface perspective.
- 3) Alarm Synchronization will work with any number of managers, each with different requirements regarding Alarm Synchronization and Alarm Reporting.
- 4) It will be possible for the manager to select the Current Alarms to be received based on the Alarm Synchronization selection criteria; these criteria must at least be able to support the criteria used in Alarm Reporting.
- 5) The manager will be able to request Alarm Synchronization on demand.
- 6) The Alarm Synchronization process will have a beginning and end that is visible on the interface.
- 7) It will be possible for a manager to invoke multiple Alarm Synchronization requests independent of any request from itself or other managers.
- 8) Alarm Synchronization shall be able to coexist with existing management information models.
- 9) The management information model will support, as an option, the cancellation (by the initiator) of a previous Alarm Synchronization request.
- 10) The Alarm Reporting parameters will not be changed when used for Alarm Synchronization.
- 11) All mandatory alarm parameters will be returned by Alarm Synchronization per matched Current Alarm.
- 12) The agent will support the reporting of one or more optional alarm parameters, as agreed upon in the Service Level Agreement. The Service Level Agreement is outside the scope of this standard.
- 13) A manager not having requested Alarm Synchronization information or not using this service, must not be

impacted by an Alarm Synchronization launched by another manager.

- 14) The agent is responsible for maintaining the Current Alarms.
- 15) Alarm Synchronization will be invoked on demand. The use of a schedule (as used with Management Operations Schedule) is not required.

7.1.4 Other Information

Fault Management systems may also want to synchronize their local version of state management information with that of an agent [11] (as an example, to determine which managed object instances are currently disabled). While this is a problem outside the scope of this standard, it can be accomplished by issuing Get request(s) utilizing the Multiple-Object Operation Service [6] on the desired state management attributes.

Alarm Synchronization does not apply to the following:

- 1) Obtain all alarms that were reported while there was a communication loss between the manager and agent system. This problem is being addressed in Recommendation X.734 Amendment 3 [14]. It defines Disseminating Log and Disseminating Queue managed objects to prevent loss of notifications during short communication losses.

In contrast to this, Alarm Synchronization must also function following an extended communications loss.

Alarm Synchronization does not return all alarms (including both the original event and the clearing event), only those alarms that have not yet been cleared. It returns the current state, not history.

- 2) An audit capability to obtain all information in the MIB specific to alarm states of the managed object instances and Current Alarms. This is a specialized form of a generic database synchronization. This problem is being addressed in Recommendation X.792 [17]. This function will only retrieve data that is actually stored in the MIB.

In contrast to this, Alarm Synchronization does not assume that Current Alarms are stored in the MIB (also see section 8.3).

7.2 Alarm Synchronization Information Model

7.2.1 Alarm Synchronization Model Overview

The Alarm Synchronization management information model is defined to overcome limitations to Current Alarm Summary Control managed object class definition criteria and reported information. Alarm Synchronization will include more alarm information than is currently contained in the Current Alarm Summary Control managed object class. This management information model uses the results iterator mechanism available with methods to identify the start and end of the reports, as shown in Figure 7-1.

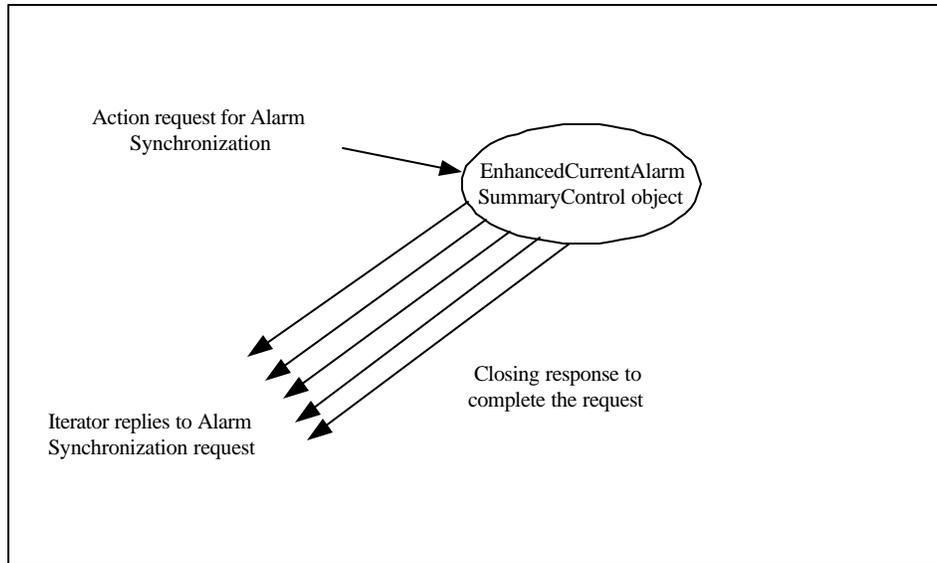


Figure 7-1. Overview of Alarm Synchronization

Alarm Synchronization action requests may be cancelled by destroying the results iterator.

7.2.2 Alarm Synchronization Managed Object Class

The following managed object class is required to meet the functional requirements specified in section 7.1.3.

7.2.2.1 EnhancedCurrentAlarmSummaryControl

The Enhanced Current Alarm Summary Control managed object class provides the functionality to perform Alarm Synchronization. Its capabilities and behavior are described throughout section 7. The definition for the EnhancedCurrentAlarmSummaryControl managed object class is shown in section 10.

7.2.3 Alarm Synchronization Inheritance Hierarchy

Figure 7-2 contains the inheritance hierarchy.

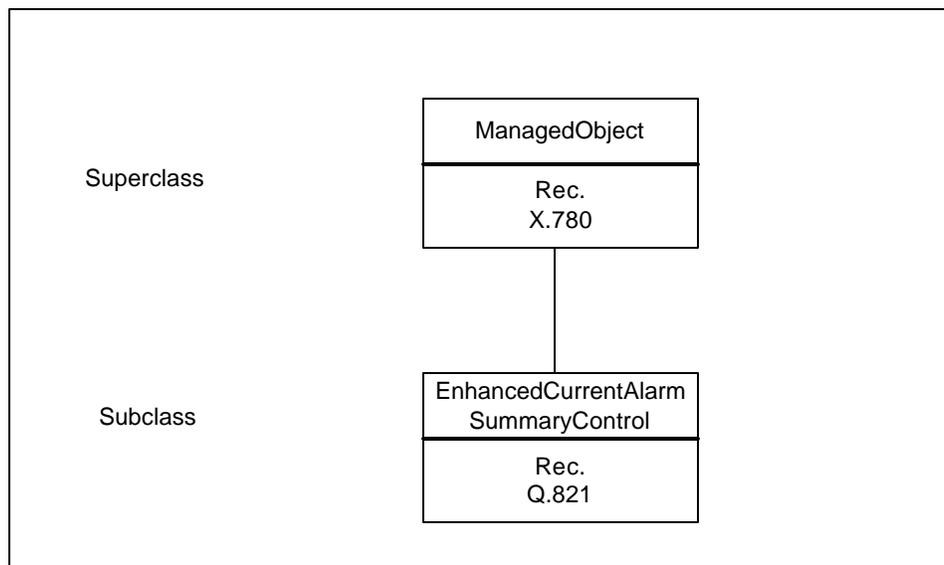


Figure 7-2. Alarm Synchronization Inheritance Hierarchy

7.2.4 Name Binding Strategies

7.2.4.1 Naming Tree Hierarchy

Figure 7-3 contains the Alarm Synchronization naming tree.

As we shall see in section 7.2.5.1, Enhanced Current Alarm Summary Control managed object instances can only retrieve alarms from managed object instances in the naming hierarchy of its immediate superior object. Any managed object instance which is not in the naming hierarchy of an Enhanced Current Alarm Summary Control's immediate superior object will not have its alarms visible via Alarm Synchronization. (Using Recommendation M.3120 [3] for an example, if a particular management information model uses Network managed objects as an immediate superior object to Managed Element managed objects and Managed Element managed objects as an immediate superior object to Enhanced Current Alarm Summary Control managed objects, then alarms from Network managed object instances would not be visible via Alarm Synchronization via those particular Enhanced Current Alarm Summary Control managed objects.)

Enhanced Current Alarm Summary Control managed object instances may be at different levels of the naming hierarchy and may have different types of immediate superior objects. Other name bindings may be defined, as require.

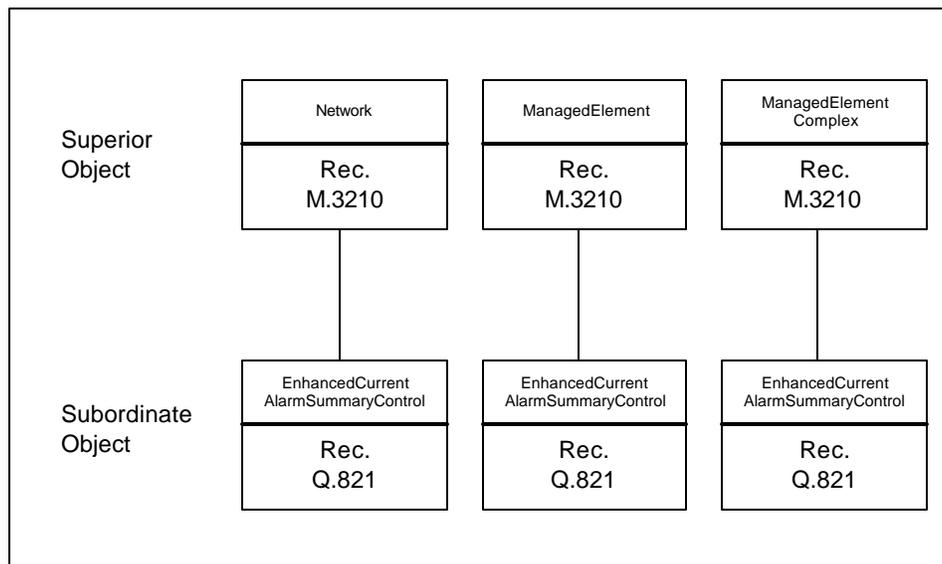


Figure 7-3. Alarm Synchronization Naming Tree Hierarchy

7.2.4.2 Object Creation and Deletion Strategy

It is the agent's responsibility to create and delete Enhanced Current Alarm Summary Control managed object instances. These instances should only be created when the agent is providing the Alarm Synchronization service. Typically, Enhanced Current Alarm Summary Control managed object instances are created upon creation of a superior managed object instance. (As an example, if the EnhancedCurrentAlarmSummaryControl_ManagedElement name binding is used, automatic creation would occur upon creation of a Managed Element managed object instance.)

The agent would use the EnhancedCurrentAlarmSummaryControlFactory to create an Enhanced Current Alarm Summary Control managed object instance.

There will typically be at most one Enhanced Current Alarm Summary Control managed object instance per immediate superior managed object instance. (As an example, if it is named relative to Managed Element and there are multiple Managed Element managed object instances, there will typically be one Enhanced Current Alarm Summary Control managed object per Managed Element instance.) The Enhanced Current Alarm Summary Control objects may occur at

different levels of the naming tree and have different types of superior objects.

7.2.4.3 EnhancedCurrentAlarmSummaryControl_ManagedElement

The name binding of Enhanced Current Alarm Summary Control to Managed Element is defined in section 10.9.

7.2.4.4 EnhancedCurrentAlarmSummaryControl_ManagedElementComplex

The name binding of Enhanced Current Alarm Summary Control to Managed Element Complex is defined in section 10.9.

7.2.4.5 EnhancedCurrentAlarmSummaryControl_Network

The name binding of Enhanced Current Alarm Summary Control to Network is defined in section 10.9.

7.2.5 Method

The Alarm Synchronization service supports the Alarm Synchronization method.

7.2.5.1 alarmSynchronization

The Alarm Synchronization method results in a download of the Current Alarm information maintained in the agent. The initiator of the request can also set selection criteria to reduce the amount of Current Alarm data returned. To reduce the amount of data returned in a single request, Alarm Synchronization utilizes an iterator (see section 7.2.5.1.3). The definition for the alarmSynchronization action is shown in section 10.

Only Current Alarms issued from the Enhanced Current Alarm Summary Control managed object instance's immediate superior or from subordinate objects of the immediate superior may be returned. As an example, if the EnhancedCurrentAlarmSummaryControl_ManagedElement name binding is used, then the Enhanced Current Alarm Summary Control managed object instance has a Managed Element managed object instance as its immediate superior. Only Current Alarms issued from this Managed Element managed object instance or from subordinate objects to this Managed Element managed object instance may be returned.

7.2.5.1.1 Alarm Synchronization Selection Criteria

The Alarm Synchronization method allows the following methods for providing the selection criteria to select which Current Alarms will be returned:

- 1) All Objects Relative To Superior – All Current Alarms issued from the Enhanced Current Alarm Summary Control managed object instance's immediate superior and its subordinate objects will be selected and returned. If no selection criteria is provided (i.e., alarmSynchronizationInfo is Null) then All Objects Relative To Superior is to be used.
- 2) Filter and scoping – This selection uses a scoping and filtering mechanism similar to that used by Alarm Reporting, as defined in Recommendation Q.816 [6].

The Base Managed Object is used as the base managed object instance for scoping. It must either consist of the Enhanced Current Alarm Summary Control managed object instance's immediate superior or a subordinate managed object instance to the Enhanced Current Alarm Summary Control managed object instance's immediate superior.

As with Recommendation Q.816, the integer-value Scope may be set to:

- 1) Base managed object only – Scope value of baseObjectOnly. Only the base managed object is included.
- 2) Whole sub-tree – Scope value of wholeSubtree. The base managed object and all managed objects contained by the base managed object are included.
- 3) Individual levels – Scope value of individualLevel along with an integer level value. Only those managed objects contained at a level equal to the integer value are included.
- 4) Base to nth level - Scope value of baseToLevel along with an integer level value. Only those managed objects contained at a level less than or equal to the value of the integer value are included.

As an example:

- 1) Scope = baseObjectOnly – Base managed object only.

- 2) Scope = wholeSubtree – Base managed object and all managed objects directly subordinate to the base managed object.
- 3) Scope = individualLevel and level = 0 – Base managed object only.
- 4) Scope = individualLevel and level = 1 – All managed objects directly subordinate to the base managed object.
- 5) Scope = baseToLevel and level = 0 – Base managed object only.
- 6) Scope = baseToLevel and level = 1 – All managed objects directly subordinate to the base managed object and including the base managed object.

Only Current Alarms issued from the selected managed object instances (via the Scope) will be further filtered.

The Criteria filter can be used to further restrict the Current Alarm selection criteria. One use of this filter is to set its value based on the Event Channel [19] used by this manager. This would then result in the same selection criteria as used in alarm reporting.

The Language parameter defines the grammar used by the Criteria filter. Only the grammar defined in Recommendation Q.816 is supported (i.e., “MOO 1.0”). Since the Recommendation Q.816 grammar is a superset of the OMG Notification Service [19] Extended TCL grammar, Event Channel Extended TCL filters may be used.

The following attributes may be specified in the filter as they apply toward Current Alarms:

- 1) Managed Object Class.
- 2) Managed Object Instance.
- 3) Event Type.
- 4) Individual notification attributes (Additional Information, Additional Text, Alarm Effect On Service, Alarming Resumed, Backed-Up Status, Back-Up Object, Correlated Notifications, Monitored Attributes, Notification Identifier, Perceived Severity, Probable Cause, Proposed Repair Actions, Specific Problems, State Change Definition, Suspect Object List, Threshold Information and Trend Indication).

Only Current Alarms issued from the selected managed object instances (via the Scope) and matching the supplied filter (via the Criteria) of a valid language (via the Language) will be returned.

- 3) Simple object list – All Current Alarms issued from managed object instances in this list will be selected and returned. Each supplied managed object instance must either contain the Enhanced Current Alarm Summary Control managed object instance's immediate superior or a subordinate managed object instance to the Enhanced Current Alarm Summary Control managed object instance's immediate superior.

Current Alarms will be selected by the methods above and returned when they have matched the supplied selection criteria. It may occur that no Current Alarms will match the supplied selection criteria.

Appendix C shows different examples of setting the selection criteria.

7.2.5.1.2 Structured Event, Event Batch And Typed Event Mapping

Recommendation X.780 [16] allows an alarm to be distributed either via a Structured Event, an Event Batch or a Typed Event. Each of these methods for distributing alarms are defined in the OMG Notification Service [19] and/or the OMG Event Service [18]. In this recommendation, all current alarms are mapped to Structured Events before they are sent via Alarm Synchronization. This section describes how to perform this mapping for an alarm distributed via Structured Event, Event Batch and Typed Event.

From Recommendation X.780, each alarm can be mapped to a Structured Event structure as shown in Figure 7-4.

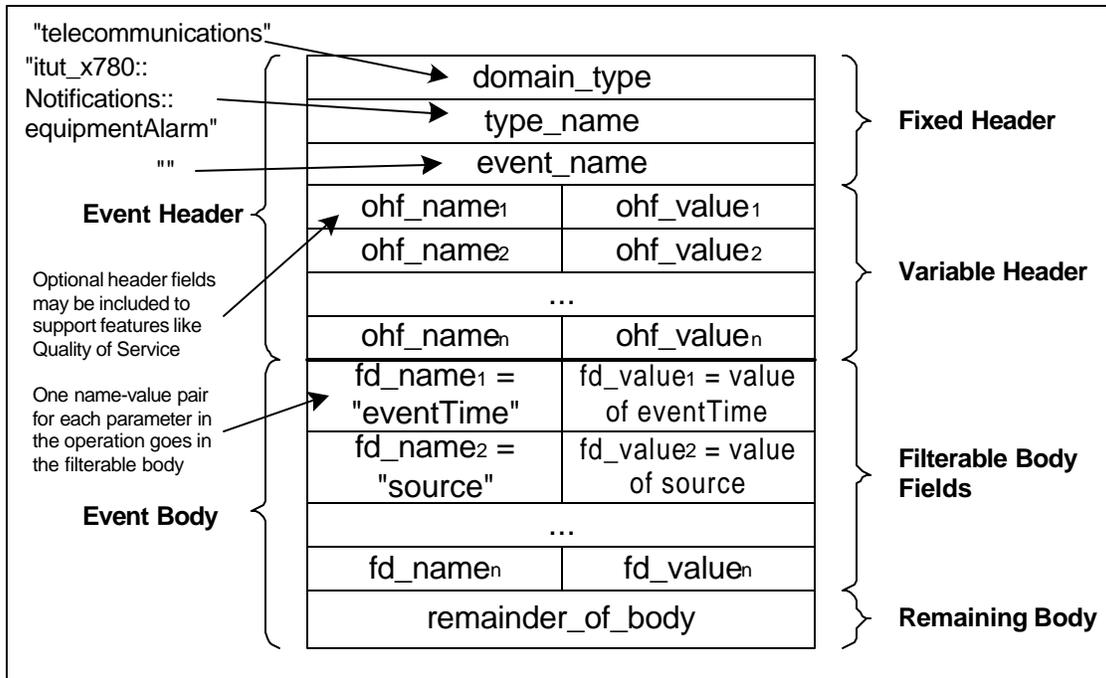


Figure 7-4. Mapping Notifications To Structured Events

An Event Batch is a sequence of Structure Events (in this context, a sequence of alarms). Alarms are mapped to Event Batches in the same way alarms are mapped to Structured Events.

The type of a typed event is dictated by mutual agreement between the agent and the manager. Alarms that are distributed via typed events are the results of executing the communicationsAlarm, environmentalAlarm, equipmentAlarm, processingErrorAlarm and qualityOfService methods in an instantiated object of type Notifications (from Recommendation X.780) (also see the TypedPushConsumer method in OMG Event Services).

OMG Notification Service defines procedures for converting Typed Events into Structured Events that are outlined in Figure 7-5.

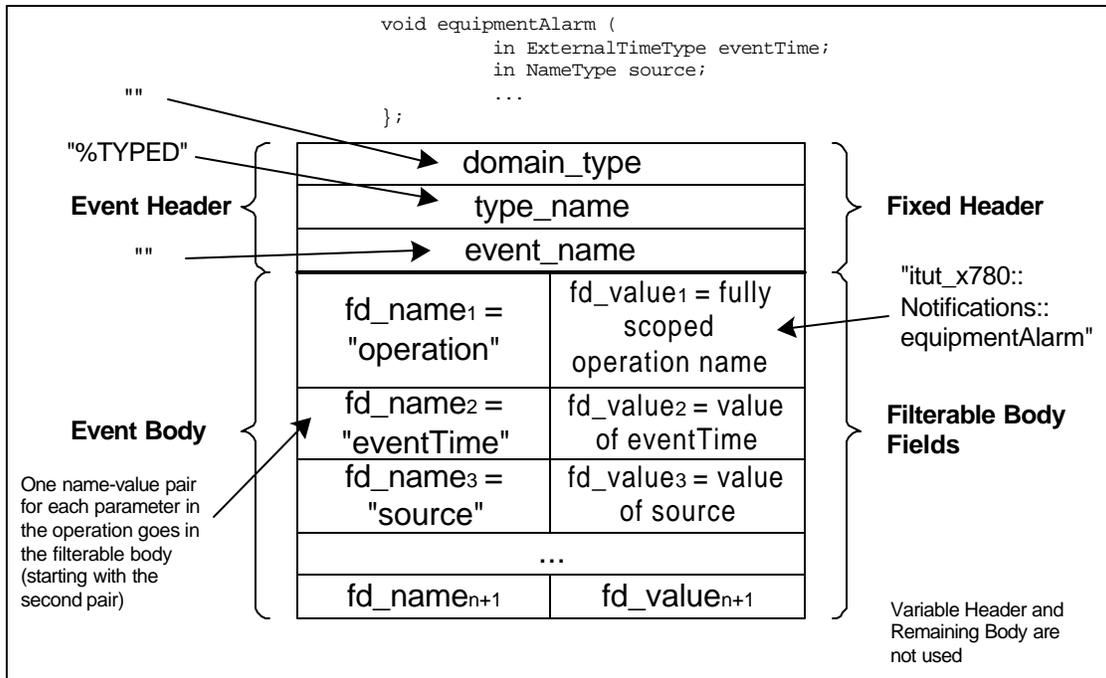


Figure 7-5. Mapping Typed Events To Structured Events

The OMG Notification Service algorithm for converting Typed Events to Structured Events is as follows:

- 1) The domain_type data member is set to the empty string.
- 2) The type_name data member is set to the "%TYPED" string.
- 3) The event_name data member is set to the empty string.
- 4) The name element of the first filterable body field name-value pair is set to the "operation" string.
- 5) The value element of the first filterable body field name-value pair is set to a string containing the fully scoped operation name. In this case, one of the following strings:
 - a) "itu_X780::Notifications::communicationsAlarm"
 - b) "itu_X780::Notifications::environmentalAlarm"
 - c) "itu_X780::Notifications::equipmentAlarm"
 - d) "itu_X780::Notifications::processingErrorAlarm"
 - e) "itu_X780::Notifications::qualityOfServiceAlarm"
- 6) The remaining name elements of the filterable body field name-value pairs are set to indicate the name of each parameter (as an example, "eventTime", "source", "sourceClass", etc.). The nth name element contains the name of the nth-1 parameter.
- 7) The remaining value elements of the filterable body field name-value pairs are set to indicate the value that was passed for each parameter (as an example, value of eventTime, value of source, value of sourceClass, etc.). The nth value element contains the value of the nth-1 parameter.

Alarm synchronization requires all current alarms to be sent via the Structured Event format. Alarm synchronization applications must either convert Typed Events to Structured Events themselves (using the above algorithm) or have OMG Notification Service do it for them (applications could create an OMG Notification Service Event Channel consumer that receives all (already converted) alarms sent to the Event Channel).

7.2.5.1.3 Alarm Synchronization Iterator

The Alarm Synchronization reply will return a sequence of Current Alarms (if any) that matches the supplied selection criteria. More than one Current Alarm may be returned per Enhanced Current Alarm Summary Control managed object instance.

The How Many input parameter indicates how many Current Alarms should be included in the first batch of responses. Zero is allowed, forcing all results to be returned through the iterator. The Results Iterator output parameter is a reference to an iterator object that may be used to retrieve additional results in batches. If all the results were returned by the Alarm Synchronization operation, the reference will be null.

It is the agent's responsibility to retrieve its Current Alarms at one moment in time (i.e., a snap shot). Since events may occur at any time, the collection of Current Alarms may be changing at any time.

The individual parameters have the same semantics as described in Recommendation X.780 [16]. The following parameters may be returned in each sequence of the action reply:

- 1) Domain Name (“telecommunications” if Structured Event or Event Batch, “” if Typed Event)
- 2) Type Name (fully scoped operation name if Structured Event or Event Batch, “%TYPED” if Typed Event)
- 3) Event Name (may be used depending on Service Level Agreement)
- 4) Operation (Typed Events only)
- 5) Event Time
- 6) Alarm Managed Object Class (from the alarm)
- 7) Alarm Managed Object Instance (from the alarm)
- 8) Notification Identifier
- 9) Correlated Notifications (optional)
- 10) Additional Text (optional)
- 11) Additional Information (optional)
- 12) Probable Cause
- 13) Specific Problems (optional)
- 14) Perceived Severity
- 15) Backed-Up Status (optional)
- 16) Back-Up Object (optional)
- 17) Trend Indication (optional)
- 18) Threshold Information (optional)
- 19) State Change Definition (optional)
- 20) Monitored Attributes (optional)
- 21) Proposed Repair Actions (optional)
- 22) Alarm Effect On Service (optional)
- 23) Alarming Resumed (optional)
- 24) Suspect Object List (optional)

As defined in Recommendation X.780 and Recommendation Q.816 [6], the Event Type will be one of the following:

- 1) “itu_X780::Notifications::communicationsAlarm”
- 2) “itu_X780::Notifications::environmentalAlarm”
- 3) “itu_X780::Notifications::equipmentAlarm”
- 4) “itu_X780::Notifications::processingErrorAlarm”
- 5) “itu_X780::Notifications::qualityOfServiceAlarm”

This standard recognizes that different agent systems will maintain different amounts of data on Current Alarm conditions. In addition, different agent systems may not maintain all of the information available in the original alarm. Therefore, this standard allows agents to optionally return less data than returned in the original alarm. As an example, if the Proposed Repair Actions parameter was not maintained by the agent for Current Alarms, then it would not be returned for Alarm Synchronization, even if it were supplied in the original alarm.

Each alarm parameter that is returned must exactly match the original alarm parameter as sent to the OMG Notification Service.

Following the initial invocation, methods in the Alarm Synchronization Data Iterator are used to complete the

accessing of the results. The Get Next method is used to access the next How Many Current Alarms. The How Many parameter must be non-zero. This method returns between 1 and How Many Current Alarms. If all of the results are returned by this invocation, FALSE is returned. The client may execute the destroy method before all of the results are retrieved (essentially canceling the Alarm Synchronization request). If TRUE is returned, then the client is either expected to make another invocation of the Get Next method or destroy the iterator. The agent will automatically destroy the iterator if all of the results are returned (i.e., FALSE is returned).

7.2.5.1.4 Alarm Synchronization Exceptions

An exception of Invalid Object Instance Error Parameter type indicates that at least one supplied Object Instance in the Object List parameter was not valid. This may typically occur if a supplied Object Instance is invalid or it is not in the Enhanced Current Alarm Summary Control managed object instance immediate superior's naming tree. The Invalid Object Instance Error Parameter will return each invalid Object Instance in the Object List. The definition for the InvalidObjectInstanceErrorParameter exception is shown in section 10.4. This will end this request.

An exception of Selection Criteria Not Supported types indicated that the agent only supports the default selection criteria of All Objects Relative To Superior and some other selection criteria was chosen. This is determined by Service Level Agreement. The definition for the SelectionCriteriaNotSupported exception is shown in section 10.4. This will end this request.

An exception of Invalid Filter type (see Recommendation Q.816 [6]) indicates the syntax of the filter is incorrect. This will end this request.

An exception of Invalid Parameter (see Recommendation Q.816 [6]) indicates that an invalid parameter was supplied. This will end this request.

An exception of Application Error type (see Recommendation X.780 [16]) indicates that some type of application error has occurred. This will end this request.

An exception of Filter Complexity Limit type (see Recommendation X.780 [16]) indicates that the supplied filter is too complex to process. This will end this request.

7.2.5.1.5 Alarm Synchronization Parameters

Parameters for the Alarm Synchronization method in the Enhanced Current Alarm Summary Control object are as shown in Table 7-1 (see Recommendation X.780 [16] for a more complete description of each parameter and Recommendation Q.816 [6] for a description of the OMG Notification Service structured events):

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Alarm Synchronization Info	M	-	One choice must be supplied
All Objects Relative To Superior	C	-	
Scoped Criteria	C	-	
Base Managed Object	C	-	
Scope	C	-	
Criteria	C	-	
Language	C	-	
Simple Object List	C	-	
How Many	M		Zero value results in empty Alarm Synchronization Data Sequence Type
Results Iterator	-	M	May be NULL if no Current Alarms
Alarm Synchronization Data Sequence Type	-	C	Of type EventBatch, which is a sequence of StructuredEvents from [19]
Event Header	-	C	
Fixed Event Header	-	C	
Event Type	-	C	
Domain Name	-	C	

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Type Name	-	C	
Event Name	-	C	
OptionalHeaderFields	-	C	Only if supplied by application
Filterable Event Body	-	C	
Property Name	-	C	“operation”, only if Typed Event
Property Value	-	C	Only if Typed Event
Event Time	-	C	Only if Typed Event
Property Name	-	C	“eventTime”
Property Value	-	C	
Event Time	-	C	Mandatory, if reply issued
Property Name	-	C	“source”
Property Value	-	C	
Alarm Managed Object Instance	-	C	Mandatory, if reply issued
Property Name	-	C	“sourceClass”
Property Value	-	C	
Alarm Managed Object Class	-	C	Mandatory, if reply issued
Property Name	-	C	“notificationIdentifier”
Property Value	-	C	
Notification Identifier	-	C	
Property Name	-	C	“correlatedNotifications”
Property Value	-	C	
Correlated Notifications	-	C	
Property Name	-	C	“additionalText”
Property Value	-	C	
Additional Text	-	C	
Property Name	-	C	“additionalInfo”
Property Value	-	C	
Additional Information	-	C	
Property Name	-	C	“probableCause”
Property Value	-	C	
Probable Cause	-	C	Mandatory, if reply issued
Property Name	-	C	“specificProblems”
Property Value	-	C	
Specific Problems	-	C	
Property Name	-	C	“perceivedSeverity”
Property Value	-	C	
Perceived Severity	-	C	Mandatory, if reply issued
Property Name	-	C	“backedUpStatus”
Property Value	-	C	
Backed-Up Status	-	C	
Property Name	-	C	“backUpObject”
Property Value	-	C	
Back-Up Object	-	C	
Property Name	-	C	“trendIndication”
Property Value	-	C	
Trend Indication	-	C	
Property Name	-	C	“thresholdInfo”
Property Value	-	C	
Threshold Information	-	C	
Property Name	-	C	“stateChangeDefinition”

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Property Value	-	C	
State Change Definition	-	C	
Property Name	-	C	“monitoredAttributes”
Property Value	-	C	
Monitored Attributes	-	C	
Property Name	-	C	“proposedRepairActions”
Property Value	-	C	
Proposed Repair Actions	-	C	
Property Name	-	C	“alarmEffectOnService”
Property Value	-	C	
Alarm Effect On Service	-	C	
Property Name	-	C	“alarmingResumed”
Property Value	-	C	
Alarming Resumed	-	C	
Property Name	-	C	“suspectObjectList”
Property Value	-	C	
Suspect Object List	-	C	
Exceptions	-	P	

Table 7-1. Alarm Synchronization Method Parameters

7.2.5.1.6 Get Next Parameters

Parameters for the Get Next method in the Alarm Synchronization Data Iterator object are as shown in Table 7-2 (see Recommendation X.780 [16] for a more complete description of each parameter):

Parameter Name	Req/Ind	Rsp/Cnf	Notes
How Many	M		Must be non-zero
Alarm Synchronization Data Sequence Type	-	C	Of type EventBatch, which is a sequence of StructuredEvents from [19]
Event Header	-	C	
Fixed Event Header	-	C	
Event Type	-	C	
Domain Name	-	C	
Type Name	-	C	
Event Name	-	C	
OptionalHeaderFields	-	C	Only if supplied by application
Filterable Event Body	-	C	
Property Name	-	C	“operation”, only if Typed Event
Property Value	-	C	Only if Typed Event
Event Time	-	C	Only if Typed Event
Property Name	-	C	“eventTime”
Property Value	-	C	
Event Time	-	C	Mandatory, if reply issued
Property Name	-	C	“source”
Property Value	-	C	
Alarm Managed Object Instance	-	C	Mandatory, if reply issued
Property Name	-	C	“sourceClass”
Property Value	-	C	
Alarm Managed Object Class	-	C	Mandatory, if reply issued
Property Name	-	C	“notificationIdentifier”

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Property Value	-	C	
Notification Identifier	-	C	
Property Name	-	C	“correlatedNotifications”
Property Value	-	C	
Correlated Notifications	-	C	
Property Name	-	C	“additionalText”
Property Value	-	C	
Additional Text	-	C	
Property Name	-	C	“additionalInfo”
Property Value	-	C	
Additional Information	-	C	
Property Name	-	C	“probableCause”
Property Value	-	C	
Probable Cause	-	C	Mandatory, if reply issued
Property Name	-	C	“specificProblems”
Property Value	-	C	
Specific Problems	-	C	
Property Name	-	C	“perceivedSeverity”
Property Value	-	C	
Perceived Severity	-	C	Mandatory, if reply issued
Property Name	-	C	“backedUpStatus”
Property Value	-	C	
Backed-Up Status	-	C	
Property Name	-	C	“backUpObject”
Property Value	-	C	
Back-Up Object	-	C	
Property Name	-	C	“trendIndication”
Property Value	-	C	
Trend Indication	-	C	
Property Name	-	C	“thresholdInfo”
Property Value	-	C	
Threshold Information	-	C	
Property Name	-	C	“stateChangeDefinition”
Property Value	-	C	
State Change Definition	-	C	
Property Name	-	C	“monitoredAttributes”
Property Value	-	C	
Monitored Attributes	-	C	
Property Name	-	C	“proposedRepairActions”
Property Value	-	C	
Proposed Repair Actions	-	C	
Property Name	-	C	“alarmEffectOnService”
Property Value	-	C	
Alarm Effect On Service	-	C	
Property Name	-	C	“alarmingResumed”
Property Value	-	C	
Alarming Resumed	-	C	
Property Name	-	C	“suspectObjectList”
Property Value	-	C	
Suspect Object List	-	C	

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Exceptions		P	

Table 7-2. Get Next Method Parameters

7.2.5.1.7 Destroy Parameters

The Destroy method in the Alarm Synchronization Data Iterator object has no parameters.

7.2.6 Notifications

The following sections describe the Alarm Synchronization event notifications. For additional information, see section 10.

7.2.6.1 Object Creation/Object Deletion

The Object Creation and Object Deletion event notifications, defined in Recommendation X.780 [16], are generated on the creation and deletion of each Enhanced Current Alarm Summary Control managed object instance. The Object Creation and Deletion Strategy is described in section 7.2.4.2.

7.3 Alarm Synchronization Functional Units

The following functional unit is defined for the management of the Alarm Synchronization function:

- 1) Alarm Synchronization functional unit - This functional unit supports the Alarm Synchronization service.

8. ALARM SYNCHRONIZATION RELATIONSHIP WITH OTHER DOCUMENTS

This specification uses the service defined in Recommendations X.780 [16] and Q.816 [6] for the notification of state changes, the creation and deletion of managed objects, the retrieval of attributes, and the notification of object creation, object deletion, and attribute value changes. Control of the reporting and logging services defined in this specification is provided by mechanisms specified in OMG Notification Service [19] and OMG Telecom Log Service [20].

8.1 Relationship With Recommendation M.3120

Recommendation M.3120 [3] defines the Current Problem List attribute. It also defines a number of managed object classes that include or optionally include the Current Problem List attribute. The Current Problem List attribute contains a set of Probable Cause and Alarm Status values. This attribute may not be included in all managed object instances that can emit alarms. It also does not include sufficient data for Alarm Synchronization.

8.2 Relationship With Recommendation X.733

Recommendation X.733 [12] defines the Alarm Reporting service. This includes the definition of what constitutes an alarm and how it is transmitted from the agent to the manager. This recommendation has been updated by Recommendation X.780 [16] for the CORBA Framework.

Alarm Synchronization totally depends on the Alarm Reporting service. It uses the alarm parameter definitions provided in Recommendation X.780. In particular, Alarm Synchronization uses the Alarm Reporting definition for alarm clearing (also see Appendix A).

The Alarm Reporting and Alarm Synchronization services may occur simultaneously. Alarms may arrive while an Alarm Synchronization action request is progressing. Issues due to the intermixing of Alarm Reporting and Alarm Synchronization data are discussed in Appendix B.

8.3 Relationship With OMG Telecom Log Service

OMG Telecom Log Service [20] defines the mechanisms for maintaining notification logs, potentially including logs of alarms in alarm records. Alarm Synchronization did not choose to use Log Records as the basis of its support for the following reasons:

- 1) Logs may be disabled or have scheduled unavailability.

- 2) Logs may become full and either wrap or halt.
- 3) Logs may be managed by managers. Log Records may be deleted from a log by a manager while they are still current. It is the responsibility of the agent to manage its collection of Current Alarms.
- 4) Logs may filter the Log Records.
- 5) Logs contain historical information. If Log Records are used, analysis is required to determine which alarms are still current and which have been cleared.

8.4 Relationship With OMG Notification Service

OMG Notification Service [19] defines the Structured Event and Event Batch structure as used for notifications. Recommendation Q.816 [6] further defines how each Structured Event field is filled for notifications. The Alarm Synchronization method uses a sequence of Structured Events (i.e., Event Batch) to return alarm information. OMG Notification Service defines how a Typed Event may be mapped to a Structured Event.

9. CONFORMANCE

There are two conformance classes: general conformance class and dependent conformance class. A system claiming to implement the elements of procedure for the services referenced by this specification shall comply with the requirements for either the general or the dependent conformance class as defined in the following clauses. The supplier of the implementation shall state the class to which the conformance is claimed.

9.1 General Conformance Class Requirement

A system claiming general conformance shall support this Stage 2 and Stage 3 Description for all managed object classes that import the management information defined in this specification.

NOTE– This is applicable to all subclasses of the support object classes defined in this specification.

9.1.1 Static Conformance

The system shall:

- a) support the role of manager or agent or both, with respect to the kernel and one or more of the functional units defined in section 5.3
- b) support the required conformance items in Recommendation X.780 [16]
- c) support the required conformance items in Recommendation Q.816 [6]
- d) when acting in the agent role, support one or more instances of the following managed object classes if such object classes are required by the supported functional units:
 - support the Alarm Severity Assignment Profile object
 - support the Current Alarm Summary Control object
 - support the Management Operations Schedule object
 - support the Enhanced Alarm Summary Control object
 - support the creation of at least one managed object of the Enhanced Current Alarm Summary Control managed object class

9.1.2 Dynamic Conformance

The system shall, in the role(s) for which conformance is claimed

- a) support the elements of procedure defined in

Recommendation X.780 [16] for the get, create, delete, set, Object creation reporting, Object deletion reporting and Attribute value change reporting services.

Recommendation Q.816 [6] for the naming, notification, Telecom Log, Factory finder, Channel finder, terminator, Multiple-Object Operation services.

9.2 Dependent Conformance Class Requirement

9.2.1 Static Conformance

The system shall:

- a) supply a system conformance statement which identifies the standardized use of this Stage 2 and Stage 3 Description
- b) support one or more instances of the relevant managed object classes required by supported functional units when acting in the agent role.

9.2.2 Dynamic Conformance

The system shall support the elements of procedure referenced by this specification as required by a standardized use of this Stage 2 and Stage 3 Description.

9.2.3 Conformance To Support Managed Object Definition

The Current Alarm Summary Control and Management Operations Schedule object classes supported by the open system shall comply with the behavior specified in section 6 and the syntax specified in section 10 of this specification.

The Channel Finder object class supported by the open system shall comply with the behavior and the syntax specified in Recommendation Q.816 [6].

The Channel Finder, Consumer Admin, Event Channel, Event Channel Factory, Filter, Filter Factory, Mapping Filter, Sequence Proxy Pull Consumer, Sequence Proxy Pull Supplier, Sequence Proxy Push Consumer, Sequence Proxy Push Supplier, Sequence Pull Consumer, Sequence Pull Supplier, Sequence Push Consumer, Sequence Push Supplier, Structured Proxy Pull Consumer, Structured Proxy Pull Supplier, Structured Proxy Push Consumer, Structured Proxy Push Supplier, Structured Pull Consumer, Structured Pull Supplier, Structured Push Consumer, Structured Push Supplier, Supplier Admin, Typed Proxy Push Consumer, Typed Proxy Push Supplier and Typed Push Consumer object classes supported by the open system shall comply with the behavior and the syntax specified in OMG Notification Service [19].

The Iterator, Notify Log, Notify Log Factory, Typed Notify Log and Typed Notify Log Factory object classes supported by the open system shall comply with the behavior and the syntax specified in OMG Telecom Log Service [20].

The Managed Element, Managed Element Complex, Network and the Alarm Severity Assignment Profile managed object class or their subclasses supported by the open system shall comply with the behavior and the syntax specified in Recommendation M.3120 [3] or the defining specification.

The Enhanced Current Alarm Summary Control managed object classes supported by the open system shall comply with the behaviour specified in section 7 and the syntax specified in section 10 of this specification.

10. RECOMMENDATION Q.821 IDL LISTING

```
#ifndef _itut_q821_idl_
#define _itut_q821_idl_

#pragma prefix "itu.int"

#include <itut_x780.idl>
#include <itut_m3120.idl>
#include <itut_q816.idl>
#include <CosNotification.idl>
```

/**

This IDL code is intended to be stored in a file named "itut_q821.idl" located in the search path used by the IDL compiler on your system

*/

/**

This module, itut_q81, contains the IDL interface definition for Recommendation Q.821 (Draft date February 2000). The IDL definitions in this file are the object interfaces.

*/

module itut_q821

{

/**

10.1 Imports

*/

/**

Types imported from Recommendation X.780

*/

typedef itut_x780::AdministrativeStateType AdministrativeStateType;

/**

Update this reference when BitStrings are actually supported in X.780

typedef itut_x780::BitStringType BitStringType;

*/

typedef itut_x780::DeletePolicyType DeletePolicyType;

typedef itut_x780::ExternalTimeType ExternalTimeType;

typedef itut_x780::GeneralizedTimeType GeneralizedTimeType;

typedef itut_x780::NameType NameType;

typedef itut_x780::NameBindingType NameBindingType;

typedef itut_x780::ObjectClassType ObjectClassType;

typedef itut_x780::OperationalStateType OperationalStateType;

typedef itut_x780::PerceivedSeverityType PerceivedSeverityType;

typedef itut_x780::ProbableCauseType ProbableCauseType;

typedef itut_x780::StartTimeType StartTimeType;

typedef itut_x780::StopTimeType StopTimeType;

typedef itut_x780::StringSetType StringSetType;

/**

Interfaces imported from Recommendation X.780

*/

typedef itut_x780::ManagedObject ManagedObject;

typedef itut_x780::ManagedObjectFactory ManagedObjectFactory;

/**

Types imported from Recommendation Q.816

*/

typedef itut_q816::FilterType FilterType;

typedef itut_q816::LanguageType LanguageType;

typedef itut_q816::ScopeType ScopeType;

/**

Types imported from Recommendation M.3120

```
*/  
  
typedef itut_m3120::AlarmStatusType AlarmStatusType;
```

```
/**  
Types imported from OMG Notification Service  
*/
```

```
typedef CosNotification::EventBatch EventBatch;
```

```
/**
```

10.2 Forward Declarations

```
*/
```

```
/**  
Interface forward declarations  
*/
```

```
interface AlarmSynchronizationDataIterator;  
interface CurrentAlarmSummaryControl;  
interface CurrentAlarmSummaryControlFactory;  
interface EnhancedCurrentAlarmSummaryControl;  
interface EnhancedCurrentAlarmSummaryControlFactory;  
interface ManagementOperationsSchedule;  
interface ManagementOperationsScheduleFactory;
```

```
/**  
valuetype forward declarations  
*/
```

```
valuetype CurrentAlarmSummaryControlValueType;  
valuetype EnhancedCurrentAlarmSummaryControlValueType;  
valuetype ManagementOperationsSchedule ValueType;
```

```
/**
```

10.3 Structures And Typedefs

```
*/
```

```
/**  
Delete this item when BitStrings are supported in X.780  
*/
```

```
typedef sequence<octet> BitStringType;
```

```
/**  
AlarmStatusList ::= SET OF AlarmStatus  
*/
```

```
typedef sequence <AlarmStatusType> AlarmStatusSetType;
```

```
/**  
ObjectList ::= SET OF ObjectListChoice  
ObjectListChoice ::= CHOICE { singleObject [1] ObjectInstance,  
rangeOfObjects [2] RangeOfObjects }
```

```
RangeOfObjects ::= SEQUENCE {
    superiorObjectName ObjectInstance,
    terminalRDNRange TerminalRDNRange }
TerminalRDNRange ::= SEQUENCE {
    attributeId OBJECT IDENTIFIER,
    firstObjectInRange INTEGER,
    lastObjectInRange INTEGER }
```

The new name for ObjectList is ObjectListSetType
*/

```
struct TerminalRDNRangeType
{
    string attributeId;
    long firstObjectInRange;
    long lastObjectInRange;
};
```

```
struct RangeOfObjectsType
{
    ManagedObject superiorObjectName;
    TerminalRDNRangeType terminalRDNRange;
};
```

```
enum ObjectListChoice
{
    singleObject,
    rangeOfObjects
};
```

/**

The rangeOfObjects may be used to specify a group of objects which are named in a contiguous manner without having to specify each instance explicitly. This mechanism may only be used to specify object instances which use integer as the final RDN of their DN. To use this mechanism, the DN of the superior object and a range of integers are specified. Each integer in the range can be concatenated with the DN of the superior object to form the DN of an indicated object.

*/

```
union ObjectListType switch (ObjectListChoice)
{
    case singleObject:
        ManagedObject singleObject;
    case rangeOfObjects:
        RangeOfObjectsType rangeOfObjects;
};
```

```
typedef sequence <ObjectListType> ObjectListSetType;
```

/**

```
PerceivedSeverityList ::= SET OF PerceivedSeverity
```

The new name for PerceivedSeverityList is PerceivedSeveritySetType

*/

```
typedef sequence <PerceivedSeverityType> PerceivedSeveritySetType;
```

```
/**
ProbableCauseList ::= SET OF ProbableCause

The new name for ProbableCauseList is ProbableCauseSetType
*/
typedef sequence <ProbableCauseType> ProbableCauseSetType;

/**
SummaryContents ::= BIT STRING { includePerceivedSeverity(0),
includeAlarmStatus(1),
includeProbableCause(2) }

The new name for SummaryContents is SummaryContentsSetType
*/
enum SummaryContentsType
{
includePerceivedSeverity,
includeAlarmStatus,
includeProbableCause
};

/**
The set must be non-empty and can't include an item more than once
*/
typedef sequence <SummaryContentsType> SummaryContentsSetType;

/**
AlarmSummaryData ::= SEQUENCE OF ObjectAlarmSummary
ObjectAlarmSummary ::= SEQUENCE{ objectOfReference ObjectOfReference,
summaryInfo SEQUENCE OF AlarmSummaryInfo }
ObjectOfReference ::= ObjectInstance
AlarmSummaryInfo ::= SEQUENCE { perceivedSeverity [0] PerceivedSeverity OPTIONAL,
alarmStatus [1] AlarmStatus OPTIONAL,
probableCause [2] ProbableCause OPTIONAL }

The new name for AlarmSummaryData is AlarmSummaryDataSeqType
*/

union PerceivedSeverityTypeOpt switch(boolean) {
case TRUE: PerceivedSeverityType perceivedSeverity;
};

union AlarmStatusTypeOpt switch(boolean) {
case TRUE: AlarmStatusType alarmStatus;
};

union ProbableCauseTypeOpt switch(boolean) {
case TRUE: ProbableCauseType probableCause;
};

/**
At least one value must be provided for AlarmSummaryInfoType
*/
```

```
struct AlarmSummaryInfoType
{
    PerceivedSeverityTypeOpt perceivedSeverityValue;
    AlarmStatusTypeOpt alarmStatusValue;
    ProbableCauseTypeOpt probableCauseValue;
};

typedef NameType ObjectOfReferenceType;
typedef sequence <AlarmSummaryInfoType> AlarmSummaryInfoSeqType;

struct ObjectAlarmSummaryType
{
    ObjectOfReferenceType objectOfReference;
    AlarmSummaryInfoSeqType summaryInfo;
};

typedef sequence <ObjectAlarmSummaryType> AlarmSummaryDataSeqType;

/**
    AffectedObjectClass ::= OBJECT IDENTIFIER
*/
typedef ObjectClassType AffectedObjectClassType;
typedef ObjectListSetType AffectedObjectInstancesType;
typedef StartTimeType BeginTimeType;
/**
    Note that the defaultEndTimeType is continual, i.e., no value. StopTimeType is too complicated of
    a type to define IDL constants
*/
typedef StopTimeType EndTimeType;

/**
    Interval ::= CHOICE { days [0] INTEGER,
                        hours [1] INTEGER,
                        minutes [2] INTEGER,
                        seconds [3] INTEGER }

    The new name for Interval is IntervalType
*/
enum IntervalChoice
{
    days,
    hours,
    minutes,
    seconds
};

union IntervalType switch (IntervalChoice)
{
    case days:
        unsigned long days;
    case hours:
        unsigned long hours;
    case minutes:

```

```
        unsigned long minutes;
    case seconds:
        unsigned long seconds;
};

/**
Correlated Record Name and Log Record Id are not supported in the CORBA model because the
application does not know the log record information before a notification is sent. Hence, these
attributes cannot be sent via the CORBA application

Suspect Object List has been moved to Recommendation M.3120
*/

/**
ScopedCriteria ::= SEQUENCE {
    baseManagedObject ObjectInstance,
    scope Scope,
    criteria CMISFilter DEFAULT and : {}
}

Note that the CORBA interface adds a language attribute as part of the MOO Services. This
application is trying to duplicate that interface

The new name for ScopedCriteria is ScopedCriteriaSeqType
*/

struct ScopedCriteriaType
{
    NameType baseManagedObject;
    /**
    Default scope is base managed object only (there is no default value for this because the
    type is too complex to represent an IDL constant)
    */
    ScopeType scope;
    /**
    Default criteria is all objects (i.e., defaultFilter in ITU-T X.780)
    */
    FilterType criteria;
    /**
    Default language is "MOO 1.0" (i.e., defaultLanguage in ITU-T X.780)
    */
    LanguageType language;
};

typedef sequence <ScopedCriteriaType> ScopedCriteriaSeqType;

/**
SimpleObjectList ::= SET OF ObjectInstance
*/
typedef sequence <ManagedObject> SimpleObjectListSetType;

/**
AlarmSynchronizationInfo ::= CHOICE {
    allObjectsRelativeToSuperior[0] NULL,
```

```
        scopedCriteria [1] ScopedCriteria,
        simpleObjectList [2] ObjectList
    }
    */

enum AlarmSynchronizationInfoChoice
{
    allObjectsRelativeToSuperior,
    scopedCriteria,
    simpleObjectList
};

union AlarmSynchronizationInfoType
switch (AlarmSynchronizationInfoChoice)
{
    case scopedCriteria:
        ScopedCriteriaSeqType scopedCriteria;
    case simpleObjectList:
        SimpleObjectListSetType simpleObjectList;
    /**
    case allObjectsRelativeToSuperior contains a NULL value
    */
};

/**
AlarmSynchronizationData ::= SEQUENCE {
    alarmManagedObjectClass ObjectClass,
    alarmManagedObjectInstance ObjectInstance,
    eventTime EventTime OPTIONAL,
    eventType EventTypeId,
    alarmInfo COMPONENTS OF AlarmInfo
}
*/

/**
In Q / CMIP, we want to keep the same data as provided as an event notification. This is why the
Alarm Synchronization Data structure was created.

In CORBA, we want to keep the same data as provided in an event notification. This is why we will
use a sequence of Notification Service structured events, called an event batch. The format of the
structured events is found in Recommendation Q.816
*/

typedef EventBatch AlarmSynchronizationDataSeqType;

/**
Note that the CORBA Framework uses the ApplicationError exception, so it will be used instead of the
InvalidBaseManagedObjectError exception
*/

/**
Error response for an invalid Object List Object Instance parameter
```

Note that the Q version returned a single managed object multiple times with multiple errors. The CORBA version will throw a single exception that may contain multiple managed objects

```
*/
typedef sequence <ManagedObject> InvalidObjectInstanceErrorSeqType;

/**
Recommendation Q.821 Correlated Record Name and Log Record Id are not supported in the CORBA
model because the application does not know the log record information before a notification is sent.
Hence, these attributes cannot be sent via the CORBA application

The parameter re-definition for SuspectObjectList is no longer needed, since the CORBA interface
doesn't differentiate between EVENT-INFO and ACTION-REPLY parameters. Thus, Suspect Object List
Action Parameter is no longer required
*/

/**
ASN.1 values not used in Recommendation Q.821 – who knows who uses them (if anyone)
*/

/**
NotificationId ::= INTEGER

ProblemData ::= SEQUENCE { identifier [0] OBJECT IDENTIFIER,
                             significance [1] BOOLEAN DEFAULT FALSE,
                             information [2] ANY DEFINED BY identifier }

StatusChange ::= SET OF SEQUENCE { statusAttributeID OBJECT IDENTIFIER,
                                     oldStatusValue [1] ANY DEFINED BY statusAttributeID OPTIONAL,
                                     newStatusValue [2] ANY DEFINED BY statusAttributeID }

CountInterval ::= SEQUENCE {
    count INTEGER,
    startTime GeneralizedTime,
    window TimeInterval }

CountWindow ::= SEQUENCE {
    count INTEGER,
    window TimeInterval }

ValueDuration ::= SEQUENCE {
    value REAL,
    duration TimeInterval }

GaugeParameters ::= CHOICE {
    up [1] SEQUENCE { high ObservedValue, low ObservedValue },
    down [2] SEQUENCE { high ObservedValue, low ObservedValue } }

Threshold ::= CHOICE {
    absoluteCount [0] INTEGER,
    countOverFixedTimeInterval [1] CountInterval,
    countOverSlidingWindow [2] CountWindow,
    valueAndDuration [3] ValueDuration,
    absoluteValue [4] REAL,
    guage [5] GaugeParameters }

```

Unclear what the semantics for these items should be, so they are not defined

*/

/**

Time interval is not used in Recommendation Q.821. At a minimum, this is used in Q.822.
TimeInterval shall be non-zero

```
TimeInterval ::= SEQUENCE {
    day [0] INTEGER (0..31) DEFAULT 0,
    hour [1] INTEGER (0..23) DEFAULT 0,
    minute [2] INTEGER (0..59) DEFAULT 0,
    second [3] INTEGER (0..59) DEFAULT 0,
    msec [4] INTEGER (0..999) DEFAULT 0 }
```

*/

struct TimeIntervalType

{

/**

Range from 0 to 31 default is 0

*/

unsigned short day;

/**

Range from 0 to 23 default is 0

*/

unsigned short hour;

/**

Range from 0 to 59 default is 0

*/

unsigned short minute;

/**

Range from 0 to 59 default is 0

*/

unsigned short second;

/**

Range from 0 to 999 default is 0

*/

unsigned short msec;

};

```
const unsigned short defaultDay = 0;
const unsigned short defaultHour = 0;
const unsigned short defaultMinute = 0;
const unsigned short defaultSecond = 0;
const unsigned short defaultMsec = 0;
```

/**

The following is the bit string to be used when specifying the functional units for alarm surveillance

```
AlarmSurveillanceFunctionalUnits ::= BIT STRING { as-kernel(0),
as-alarm-retrieval(1),
as-basic-arc(2),
as-enhanced-arc(3),
as-cur-alm-sum-reporting(4),
as-basic-mos(5),
```

```
as-enhanced-mos(6),
as-cur-alm-sum-control(7),
as-cur-alm-sum-retrieval(8),
as-basic-log-control(9),
as-enhanced-log-control(10),
as-alarm-deletion(11),
as-alarm-event-criteria(12),
as-alarm-indication(13)
as-alarm-synch(14)
as-alarm-synch-cancel(15)}
```

It isn't yet clear how to define the constants. as-heartbeat has been added and as-alarm-synch-cancel is no longer supported. The Functional Unit bit strings are currently not defined in CORBA.
*/

```
/**
```

10.4 Exceptions

```
*/
```

```
/**
"-- see section 7.2.5.1.4 on model --"
*/
exception InvalidObjectInstanceErrorParameter {
    InvalidObjectInstanceErrorSeqType objectInstanceSequence;
};

/**
"-- see section 7.2.5.1.4 on model --"
*/
exception SelectionCriteriaNotSupported {};

/*
Used for managed object instances where the managementOperationsScheduleOperationalStatePkg
package isn't used
*/
exception NOmanagementOperationsScheduleOperationalStatePkg {};
```

```
/**
```

10.5 Current Alarm Summary Control

```
*/
```

```
/**
This valuetype is used to retrieve all attributes
*/

valuetype CurrentAlarmSummaryControlValueType : itut_x780::ManagedObjectValueType {
    public AlarmStatusSetType alarmStatusList;
        // GET-REPLACE, ADD-REMOVE
    public ObjectListSetType objectList;
        // GET-REPLACE, ADD-REMOVE
    public PerceivedSeveritySetType perceivedSeverityList;
        // GET-REPLACE, ADD-REMOVE
    public ProbableCauseSetType probableCauseList;
```

```
        // GET-REPLACE, ADD-REMOVE  
}; // valuetype CurrentAlarmSummaryControlValueType
```

```
/**  
"-- see section 6.6.2.1 --";  
*/
```

```
/**  
Current Alarm Summary Control managed object
```

The Current Alarm Summary Control and Management Operations Schedule managed objects have been supplied for completeness. It is expected that most CORBA applications would use the Enhanced Current Alarm Summary Control managed object instead of these two managed objects.

```
*/
```

```
interface CurrentAlarmSummaryControl : ManagedObject  
{
```

```
    /**  
    "-- see section 6.6.2.1 a) --"
```

```
    alarmStatusList GET-REPLACE, ADD-REMOVE  
    */
```

```
    AlarmStatusSetType alarmStatusListGet ()  
        raises (itut_x780::ApplicationError);
```

```
    void alarmStatusListSet  
        (in AlarmStatusSetType alarmStatusList)  
        raises (itut_x780::ApplicationError);
```

```
    void alarmStatusListAdd  
        (in AlarmStatusSetType alarmStatusList)  
        raises (itut_x780::ApplicationError);
```

```
    void alarmStatusListRemove  
        (in AlarmStatusSetType alarmStatusList)  
        raises (itut_x780::ApplicationError);
```

```
    /**  
    "-- see section 6.6.2.1 b) --";
```

```
    objectList GET-REPLACE, ADD-REMOVE  
    */
```

```
    ObjectListSetType objectListGet ()  
        raises (itut_x780::ApplicationError);
```

```
    void objectListSet  
        (in ObjectListSetType objectList)  
        raises (itut_x780::ApplicationError);
```

```
    void objectListAdd  
        (in ObjectListSetType objectList)  
        raises (itut_x780::ApplicationError);
```

```
void objectListRemove
    (in ObjectListSetType objectList)
    raises (itut_x780::ApplicationError);

/**
"-- see section 6.6.2.1 c) --";

perceivedSeverityList GET-REPLACE, ADD-REMOVE
*/

PerceivedSeveritySetType perceivedSeverityListGet ()
    raises (itut_x780::ApplicationError);

void perceivedSeverityListSet
    (in PerceivedSeveritySetType perceivedSeverityList)
    raises (itut_x780::ApplicationError);

void perceivedSeverityListAdd
    (in PerceivedSeveritySetType perceivedSeverityList)
    raises (itut_x780::ApplicationError);

void perceivedSeverityListRemove
    (in PerceivedSeveritySetType perceivedSeverityList)
    raises (itut_x780::ApplicationError);

/**
"-- see section 6.6.2.1 d) --"

probableCauseList GET-REPLACE, ADD-REMOVE
*/

ProbableCauseSetType probableCauseListGet ()
    raises (itut_x780::ApplicationError);

void probableCauseListSet
    (in ProbableCauseSetType probableCauseList)
    raises (itut_x780::ApplicationError);

void probableCauseListAdd
    (in ProbableCauseSetType probableCauseList)
    raises (itut_x780::ApplicationError);

void probableCauseListRemove
    (in ProbableCauseSetType probableCauseList)
    raises (itut_x780::ApplicationError);

/**
"-- see section 6.7.10.1 --"

@param summaryContents                Indicates whether Perceived Severity, Alarm Status and/or
                                        Probable Cause are included in the results
@param alarmSummaryData                Sequences of alarm summaries including Perceived
                                        Severity, Alarm Status and/or Probable Cause, depending
                                        on summary contents
```

```
*/

void retrieveCurrentAlarmSummary
    (in SummaryContentsSetType summaryContents,
     out AlarmSummaryDataSeqType alarmSummaryData)
    raises (itut_x780::ApplicationError);

MANDATORY_NOTIFICATION
    (itut_x780::Notifications, currentAlarmSummaryReport)

}; // interface CurrentAlarmSummaryControl

/**
Creation and Deletion for Current Alarm Summary Control
*/

interface CurrentAlarmSummaryControlFactory : ManagedObjectFactory
{
    CurrentAlarmSummaryControl create
        (in NameBindingType nameBinding,
         in ManagedObject superior,
         in string name, // no auto-naming, cannot be null
         in StringSetType packages,
         in AlarmStatusSetType alarmStatusList,
          // GET-REPLACE, ADD-REMOVE
         in ObjectListSetType objectList,
          // GET-REPLACE, ADD-REMOVE
         in PerceivedSeveritySetType perceivedSeverityList,
          // GET-REPLACE, ADD-REMOVE
         in ProbableCauseSetType probableCauseList)
          // GET-REPLACE, ADD-REMOVE
        raises (itut_x780::ApplicationError,
               itut_x780::CreateError);

}; // interface CurrentAlarmSummaryControlFactory

/**
10.6 Management Operations Schedule
*/

/**
This valuetype is used to retrieve all attributes
*/

valuetype ManagementOperationsSchedule ValueType : itut_x780::ManagedObjectValueType {
    public AdministrativeStateType administrativeState;
        // GET-REPLACE
    public AffectedObjectClassType affectedObjectClass;
        // GET-REPLACE
    public AffectedObjectInstancesType affectedObjectInstances;
        // GET-REPLACE
    public BeginTimeType beginTime;
        // GET-REPLACE
    public EndTimeType endTime;
}
```

```
        // GET-REPLACE
public IntervalType interval;
        // GET-REPLACE
public OperationalStateType operationalState;
        // conditional
        // managementOperationsScheduleOperationalStatePkg
        // GET
}; // valuetype ManagementOperationsSchedule ValueType
```

```
/**
"-- see section 6.6.2.2 --";
*/
```

```
/**
Management Operations Schedule managed object
```

Destination Address is not needed in a CORBA world. In the CORBA world, what channel is to be used is outside the scope of an object. Thus, a managed object cannot send a message from one channel or another

```
interface ManagementOperationsSchedule : ManagedObject
{
    /**
    "-- see section 6.6.2.2 a) --"

    administrativeState GET-REPLACE
    */

    AdministrativeStateType administrativeStateGet ()
        raises (itut_x780::ApplicationError);

    void administrativeStateSet
        (in AdministrativeStateType administrativeState)
        raises (itut_x780::ApplicationError);

    /**
    "-- see section 6.6.2.2 b) --"

    affectedObjectClass GET-REPLACE
    */

    AffectedObjectClassType affectedObjectClassGet ()
        raises (itut_x780::ApplicationError);

    void affectedObjectClassSet
        (in AffectedObjectClassType affectedObjectClass)
        raises (itut_x780::ApplicationError);

    /**
    "-- see section 6.6.2.2 c) --"

    affectedObjectInstances GET-REPLACE
    */
```

```
AffectedObjectInstancesType affectedObjectInstancesGet ()
    raises (itut_x780::ApplicationError);

void affectedObjectInstancesSet
    (in AffectedObjectInstancesType affectedObjectInstances)
    raises (itut_x780::ApplicationError);

/**
"-- see section 6.6.2.2 d) --"
first activation at begin time, if present, or else when schedule is created

beginTime GET-REPLACE
*/

BeginTimeType beginTimeGet ()
    raises (itut_x780::ApplicationError);

void beginTimeSet
    (in BeginTimeType beginTime)
    raises (itut_x780::ApplicationError);

/**
"-- see section 6.6.2.2 e) --"

endTime GET-REPLACE
*/

EndTimeType endTimeGet ()
    raises (itut_x780::ApplicationError);

void endTimeSet
    (in EndTimeType endTime)
    raises (itut_x780::ApplicationError,
           itut_q816::InvalidParameter);

/**
The default EndTimeType is continual
*/

void endTimeSetDefault ()
    raises (itut_x780::ApplicationError,
           itut_q816::InvalidParameter);

/**
"-- see section 6.6.2.2 f) --"

interval GET-REPLACE
*/

IntervalType intervalGet ()
    raises (itut_x780::ApplicationError);

void intervalSet
    (in IntervalType interval)
    raises (itut_x780::ApplicationError);
```

```
/**
Conditional Package managementOperationsScheduleOperationalStatePkg PRESENT IF "an
instance supports it.";

"-- see section 6.6.2.2 g) --"

operationalState GET
*/

OperationalStateType operationalStateGet ()
    raises (itut_x780::ApplicationError,
           NOmanagementOperationsScheduleOperationalStatePkg);

}; // interface ManagementOperationsSchedule

/**
Creation and Deletion for Management Operations Schedule
*/

interface ManagementOperationsScheduleFactory : ManagedObjectFactory
{
    ManagementOperationsSchedule create
        (in NameBindingType nameBinding,
         in ManagedObject superior,
         in string name, // no auto-naming, cannot be null
         in StringSetType packages,
         in AdministrativeStateType administrativeState,
          // GET-REPLACE
         in AffectedObjectClassType affectedObjectClass,
          // GET-REPLACE
         in AffectedObjectInstancesType affectedObjectInstances,
          // GET-REPLACE
         in BeginTimeType beginTime,
          // GET-REPLACE
         in EndTimeType endTime,
          // GET-REPLACE
         in IntervalType interval,
          // GET-REPLACE
         in OperationalStateType operationalState)
        // conditional
        // managementOperationsScheduleOperationalStatePkg
        // GET
        raises (itut_x780::ApplicationError,
               itut_x780::CreateError);

}; // interface ManagementOperationsScheduleFactory

/**
10.7 Enhanced Current Alarm Summary Control
*/
```

```
/**
Note that the Enhanced Current Alarm Summary Control managed object has no attributes, the naming attribute is
```

```
no longer needed
*/

valuetype EnhancedCurrentAlarmSummaryControlValueType : itut_x780::ManagedObjectValueType {
}; // valuetype EnhancedCurrentAlarmSummaryControlValueType

/**
"-- see section 7.2.2 on model --"
*/

/**
Enhanced Current Alarm Summary Control managed object
*/

interface EnhancedCurrentAlarmSummaryControl : ManagedObject
{
    /**
    "-- see section 7.2.5 on model --"
    */

    /**
    The Alarm Synchronization method is used to return current alarms to the manager.

    @param alarmSynchronizationInfo          What managed object alarms are we interested in? If this is
                                              Null, then assume All Managed Objects Relative
                                              To Superior

    @param howMany                          Maximum number of alarms for which results
                                              should be returned in first batch, 0 indicates all
                                              results returned via iterator

    @param resultsIterator                  A reference to an iterator containing the results.
                                              This could be NULL if no results are returned

    @param AlarmSynchronizationDataSeqType Sequence of current alarms, if howMany is non-
                                              zero

    AlarmSynchronizationDataSeqType alarmSynchronization
    (in AlarmSynchronizationInfoType alarmSynchronizationInfo,
    in unsigned short howMany,
    out AlarmSynchronizationDataIterator resultsIterator)
    raises (itut_x780::ApplicationError,
           InvalidObjectInstanceErrorParameter,
           SelectionCriteriaNotSupported,
           itut_q816::InvalidFilter,
           itut_q816::InvalidParameter,
           itut_q816::FilterComplexityLimit);

    /**
    The concept of Cancel Alarm Synchronization is no longer required. First, you can't cancel an alarm
    synchronization call that is currently under progress. Secondly, the application can destroy an iterator
    once it has a returned.

    This also removes the need for: noSuchInvokeIdErrorParameter and
    canceledAlarmSynchronizationParameter
    */
}
```

```
MANDATORY_NOTIFICATION
    (itut_x780::Notifications, objectCreation)
MANDATORY_NOTIFICATION
    (itut_x780::Notifications, objectDeletion)

}; // interface EnhancedCurrentAlarmSummaryControl

/**
The Alarm Synchronization Data Iterator allows the alarmSynchronization action to be returned in multiple calls.
This is necessary since a very large number of alarms could be returned and this could be larger than CORBA
allows to be returned in one call

The agent system controls the life-cycle of the iterator. However, a destroy operation is provided to handle
the case where the manager wants to stop the iteration procedure before reaching the last iteration.

"-- see section 7.2.5.1.3 on model --"
*/

interface AlarmSynchronizationDataIterator
{
    /**
    This method is used to return the next how many current alarms via Alarm Synchronization. This
    method returns between 1 and "howMany" current alarms. The agent may return less than
    "howMany" items even if there are more items to send. "howMany" must be non-zero. Return
    TRUE if there are more current alarms to return. Return FALSE if there are no more current alarms to
    be returned. Note that the agent may both provide the last current alarm in the list and also indicate
    FALSE for completion.

    If FALSE is returned, the agent will automatically destroy the iterator.

    @param howMany                Maximum number of alarms for which results
                                should be returned in this batch, must be non-zero
    @param AlarmSynchronizationDataSeqType Sequence of current alarms, see section 7.2.5.1.2
                                on how to map alarms to fit the Structure Event
                                structure
    */

    boolean getNext
        (in unsigned short howMany,
         out AlarmSynchronizationDataSeqType currentAlarms)
        raises (itut_q816::InvalidParameter,
               itut_x780::ApplicationError);

    /**
    This method is used to destroy the iterator and release its resources.
    */

    void destroy ();

}; // interface AlarmSynchronizationDataIterator

/**
"-- see section 7.2.4.2 on model --"
*/
```

```
*/  
  
/**  
Creation and Deletion for Enhanced Current Alarm Summary Control  
*/  
  
interface EnhancedCurrentAlarmSummaryControlFactory : ManagedObjectFactory  
{  
    EnhancedCurrentAlarmSummaryControl create  
        (in NameBindingType nameBinding,  
         in ManagedObject superior,  
         in string name, // no auto-naming, cannot be null  
         in StringSetType packages)  
        raises (itut_x780::ApplicationError,  
              itut_x780::CreateError);  
  
}; // interface EnhancedCurrentAlarmSummaryControl  
  
/**  
10.8 Notifications  
*/  
  
/**  
"-- see section 6.7.6.1 --"  
  
@param alarmSummaryData    Sequences of alarm summaries including Perceived Severity, Alarm Status  
                             and/or Probable Cause, depending on summary contents  
*/  
  
interface Notifications {  
    void currentAlarmSummaryReport  
        (in AlarmSummaryDataSeqType alarmSummaryData);  
};  
  
/**  
Notification constants  
*/  
  
const string currentAlarmSummaryReportTypeName =  
    "itu_q821::Notifications::currentAlarmSummaryReport";  
const string alarmSummaryDataName = "alarmSummaryData";  
  
/**  
10.9 Name Binding  
*/  
  
module NameBinding  
{  
    /**  
    This name binding is used to name the Current Alarm Summary Control object to a Managed Element  
    object.  
    */  
}
```

```
module CurrentAlarmSummaryControl_ManagedElement
{
    const string superiorClass = "itut_m3120::ManagedElement";
    const boolean superiorSubclassesAllowed = TRUE;
    const string subordinateClass = "itut_q821::CurrentAlarmSummaryControl";
    const boolean subordinateSubclassesAllowed = TRUE;
    const boolean managerCreatesAllowed = TRUE;
    const DeletePolicyType deletePolicy = itu_x780::deleteContainedObjects;
    const string kind = "CurrentAlarmSummaryControl";

}; // module CurrentAlarmSummaryControl_ManagedElement

/**
This name binding is used to name the Management Operations Schedule object to a Managed Element
object.
*/

module ManagementOperationsSchedule_ManagedElement
{
    const string superiorClass = "itut_m3120::ManagedElement";
    const boolean superiorSubclassesAllowed = TRUE;
    const string subordinateClass = "itut_q821::ManagementOperationsSchedule";
    const boolean subordinateSubclassesAllowed = TRUE;
    const boolean managerCreatesAllowed = TRUE;
    const DeletePolicyType deletePolicy = itu_x780::deleteContainedObjects;
    const string kind = "ManagementOperationsSchedule";

}; // module ManagementOperationsSchedule_ManagedElement

/**
This name binding is used to name the Enhanced Current Alarm Summary Control object to a Managed
Element object. This object is not created or deleted by system management protocol.
*/

module EnhancedCurrentAlarmSummaryControl_ManagedElement
{
    const string superiorClass = "itut_m3120::ManagedElement";
    const boolean superiorSubclassesAllowed = TRUE;
    const string subordinateClass = "itut_q821::EnhancedCurrentAlarmSummaryControl";
    const boolean subordinateSubclassesAllowed = TRUE;
    const boolean managerCreatesAllowed = FALSE;
    const DeletePolicyType deletePolicy = itu_x780::notDeletable;
    const string kind = "EnhancedCurrentAlarmSummaryControl";

}; // module EnhancedCurrentAlarmSummaryControl_ManagedElement

/**
This name binding is used to name the Enhanced Current Alarm Summary Control object to a Managed
Element Complex object. This object is not created or deleted by system management protocol.
*/

module EnhancedCurrentAlarmSummaryControl_ManagedElementComplex
{
    const string superiorClass = "itut_m3120::ManagedElementComplex";
```

```
const boolean superiorSubclassesAllowed = TRUE;
const string subordinateClass = "itut_q821::EnhancedCurrentAlarmSummaryControl";
const boolean subordinateSubclassesAllowed = TRUE;
const boolean managerCreatesAllowed = FALSE;
const DeletePolicyType deletePolicy = itu_x780::notDeletable;
const string kind = "EnhancedCurrentAlarmSummaryControl";

}; // module EnhancedCurrentAlarmSummaryControl_ManagedElementComplex

/**
This name binding is used to name the Enhanced Current Alarm Summary Control object to a Network
object. This object is not created or deleted by system management protocol.
*/

module EnhancedCurrentAlarmSummaryControl_Network
{
    const string superiorClass = "itut_m3120::Network";
    const boolean superiorSubclassesAllowed = TRUE;
    const string subordinateClass = "itut_q821::EnhancedCurrentAlarmSummaryControl";
    const boolean subordinateSubclassesAllowed = TRUE;
    const boolean managerCreatesAllowed = FALSE;
    const DeletePolicyType deletePolicy = itu_x780::notDeletable;
    const string kind = "EnhancedCurrentAlarmSummaryControl";

}; // module EnhancedCurrentAlarmSummaryControl_Network

}; // module NameBinding

}; // module itut_q821

#endif // _itut_q821_idl_
```

APPENDIX A - Integrating Alarm Synchronization With Alarm Clearing

Alarm Clearing Clarified

According to Recommendation X.733 [12], the use of clearing alarms is optional. When they are provided, alarms are cleared based on their Perceived Severity value. A Perceived Severity value of “Cleared” can clear zero, one or more previously reported alarms.

Alarms are cleared based on the value of the following alarm parameters:

- 1) Managed Object Class.
- 2) Managed Object Instance.
- 3) Event Type.
- 4) Probable Cause.
- 5) Specific Problems (when provided). An empty Specific Problems is to be treated equivalent to a Specific Problems parameter that is not provided.
- 6) Correlated Notifications (when provided). An empty Correlated Notifications is to be treated equivalent to a Correlated Notifications parameter that is not provided.

Multiple collections of alarms may be cleared based on the Correlated Notifications attribute. The Correlated Notifications attribute contains a set of Notification Identifiers for which this new alarm is correlated. (It also will include Managed Object Instances when the alarm containing the Notification Identifier was from a different Managed Object Instance than used in this alarm.) Alarms may be correlated with alarms from any managed object instance.

All Notification Identifiers contained in the Correlated Notifications attribute must already have been used in the Notification Identifier attribute of a previously sent alarm. Notification Identifier values must be unique across all notifications of a particular managed object class throughout the time the correlation is significant.

Alarms are cleared based on the following cases:

- 1) Specific Problems and Correlated Notifications not provided (or empty) in the clearing alarm – Clear all alarms that exactly match Managed Object Class, Managed Object Instance, Event Type and Probable Cause. Alarms are cleared irrespective of their Specific Problems or Correlated Notifications values.
- 2) Specific Problems is provided (and non-empty) and Correlated Notifications is not provided (or empty) in the clearing alarm – Clear all alarms that exactly match Managed Object Class, Managed Object Instance, Event Type, Probable Cause and Specific Problems. Also, clear all alarms that exactly match Managed Object Class, Managed Object Instance, Event Type and Probable Cause and match a non-empty subset of Specific Problems. Alarms are cleared irrespective of their Correlated Notifications values.
- 3) Specific Problems is not provided (or empty) and Correlated Notifications is provided (and non-empty) in the clearing alarm – Clear all alarms in the Correlated Notifications set. No other alarms are cleared by this alarm. Note that this may result in alarms from other managed object classes being cleared.
- 4) Specific Problems and Correlated Notifications are provided (and non-empty) in the clearing alarm – Clear all alarms in the Correlated Notifications set. No other alarms are cleared by this alarm. Note that this may result in alarms from other managed object classes being cleared.

Some systems choose to clear alarms one-for-one, to reduce this complexity.

Since alarm correlation is integrated with alarm clearing, it is important that the agent clears the old alarm conditions and resends them as new alarms (with updated Correlated Notifications information) when it changes its alarm correlations.

Alarm clearing may be progressing while Alarm Synchronization data is arriving. See Appendix B for a discussion of these issues.

Alarm Clearing Examples

The following examples are used to demonstrate the clearing of alarms. In these examples, the alarms have arrived in ascending order and are still pending. “-” is used to indicate the optional field wasn’t provided. The letters A, B, etc. are used to indicate different values for that particular attribute type. As an example, MOC-A (Managed Object Class A) is different from MOC-B.

Alarm	1	2	3	4	5	6	7	8	9
Managed Object Class	MOC-A	MOC-A	MOC-A	MOC-A	MOC-A	MOC-A	MOC-B	MOC-B	MOC-A
Managed Object Instance	MOI-A	MOI-A	MOI-A	MOI-A	MOI-B	MOI-B	MOI-C	MOI-D	MOI-A
Event Type	ET-A	ET-A	ET-A	ET-A	ET-A	ET-A	ET-A	ET-A	ET-A
Probable Cause	PC-A	PC-A	PC-A	PC-A	PC-A	PC-A	PC-A	PC-A	PC-A
Specific Problems	-	-	{SP-A}	{SP-A} {SP-C}	-	-	-	{SP-B}	-
Perceived Severity	Critical	Major	Minor	Minor	Critical	Major	Critical	Critical	Critical
Notification Identifier	-	-	-	-	54	55	54	55	56
Correlated Notifications	-	-	-	-	-	{54}	-	{55} {54,MOI-C}	{56} {54,MOI-B} {55,MOI-B} {54,MOI-C} {55,MOI-D}

Table A.1. Sample Non-Cleared Alarms

The following examples list clearing alarms and which of the alarms in Table A.1 would be cleared if that alarm were the next alarm to arrive.

Individual examples:

- a) Managed Object Class = MOC-A.
Managed Object Instance = MOI-A.
Event Type = ET-A.
Probable Cause = PC-B.
Specific Problems = <not supplied>.
Perceived Severity = Cleared.
Correlated Notifications = <not supplied>.
Resulting Alarms Cleared = none (Case #1). [Different Probable Cause]
- b) Managed Object Class = MOC-A.
Managed Object Instance = MOI-A.
Event Type = ET-A.
Probable Cause = PC-A.
Specific Problems = <not supplied>.
Perceived Severity = Cleared.
Correlated Notifications = <not supplied>.

- Resulting Alarms Cleared = 1, 2, 3, 4 and 9 (Case #1).
- c) Managed Object Class = MOC-A.
Managed Object Instance = MOI-A.
Event Type = ET-A.
Probable Cause = PC-A.
Specific Problems = {SP-A}.
Perceived Severity = Cleared.
Correlated Notifications = <not supplied>.
Resulting Alarms Cleared = 3 (Case #2).
- d) Managed Object Class = MOC-A.
Managed Object Instance = MOI-A.
Event Type = ET-A.
Probable Cause = PC-A.
Specific Problems = {SP-A}{SP-C}.
Perceived Severity = Cleared.
Correlated Notifications = <not supplied>.
Resulting Alarms Cleared = 3 and 4 (Case #2).
- e) Managed Object Class = MOC-A.
Managed Object Instance = MOI-B.
Event Type = ET-A.
Probable Cause = PC-A.
Specific Problems = <not supplied>.
Perceived Severity = Cleared.
Correlated Notifications = <not supplied>.
Resulting Alarms Cleared = 5 and 6 (Case #1).
- f) Managed Object Class = MOC-A.
Managed Object Instance = MOI-B.
Event Type = ET-A.
Probable Cause = PC-A.
Specific Problems = <not supplied>.
Perceived Severity = Cleared.
Correlated Notifications = {54}.
Resulting Alarms Cleared = 5 (Case #3).
- g) Managed Object Class = MOC-B.
Managed Object Instance = MOI-D.
Event Type = ET-A.
Probable Cause = PC-A.
Specific Problems = {SP-B}.
Perceived Severity = Cleared.
Correlated Notifications = {55}{54,MOI-C}.
Resulting Alarms Cleared = 7 and 8 (Case #4).
- h) Managed Object Class = MOC-A.
Managed Object Instance = MOI-A.
Event Type = ET-A.
Probable Cause = PC-A.
Specific Problems = <not supplied>.
Perceived Severity = Cleared.
Correlated Notifications = {56}{54,MOI-B}{55,MOI-B}{54,MOI-C}{55,MOI-D}.
Resulting Alarms Cleared = 5, 6, 7, 8 and 9 (Case #3).

APPENDIX B - Alarm Synchronization Race Condition

Overview

Alarm Reporting [12] alarms may arrive while Alarm Synchronization alarms are being retrieved. This means that new alarms can be interleaved with Alarm Synchronization Current Alarms. The new alarms may either be a fault or a clearing of a fault. Managers that process Alarm Reporting alarms and Alarm Synchronization alarms sequentially may arrive with a race condition when Alarm Reporting clear alarms arrive before the matching Alarm Synchronization alarms. This is a race condition because the end results will depend on the order of the arriving alarms. This is probably best shown through an example.

Example

Let's go through a simple example. The manager has initiated an Alarm Synchronization request. Suppose the agent has the four Current Alarms as shown in Table B.1. Also suppose the agent has a notification which will clear one of the Current Alarms. We know that asynchronous event notifications can still arrive any time during an Alarm Synchronization request. Suppose these alarms arrive in the following order:

Order	1	2	3	4	5
Type	Alarm Synch	Alarm Synch	Alarm Synch	Alarm Reporting	Alarm Synch
Managed Object Class	MOC-A	MOC-B	MOC-C	MOC-C	MOC-D
Managed Object Instance	MOI-A	MOI-B	MOI-C	MOI-C	MOI-D
Event Type	ET-A	ET-A	ET-A	ET-A	ET-A
Probable Cause	PC-A	PC-A	PC-A	PC-A	PC-A
Perceived Severity	Critical	Major	Minor	Cleared	Major

Table B.1. Race Condition Example #1

If the manager processes these requests in order from 1 through 5, the Alarm Reporting alarm will clear the Alarm Synchronization alarm that arrived earlier. The end result is that alarms for MOC-A, MOC-B and MOC-D still exist (which is correct).

If we change the order slightly, as in Table B.2, we could end up with the following result:

Order	1	2	4	4	5
Type	Alarm Synch	Alarm Synch	Alarm Reporting	Alarm Synch	Alarm Synch
Managed Object Class	MOC-A	MOC-B	MOC-C	MOC-C	MOC-D
Managed Object Instance	MOI-A	MOI-B	MOI-C	MOI-C	MOI-D
Event Type	ET-A	ET-A	ET-A	ET-A	ET-A
Probable Cause	PC-A	PC-A	PC-A	PC-A	PC-A
Perceived Severity	Critical	Major	Cleared	Minor	Major

Table B.2. Race Condition Example #2

If the manager processes these requests in order, the Alarm Reporting alarm will not clear an Alarm Synchronization alarm, since it hasn't already arrived. The end result is that alarms for MOC-A, MOC-B, MOC-C and MOC-D still exist (which is incorrect, the end result should be alarms for MOC-A, MOC-B and MOC-D).

This is even more complicated when Correlated Notifications and how they clear alarms are taken into effect (see Appendix A).

Suggestions On Handling This Race Condition

New Alarm Reporting alarms still need to be processed while Alarm Synchronization is occurring. It typically is not prudent to ignore critical alarms until an Alarm Synchronization request has completed.

The handling of this race condition must be handled by the manager. The agent is sending a snap shot of its Current Alarms and sending asynchronous events when they occur.

The arriving of clearing alarms that are outside the scope of the Alarm Synchronization selection criteria will not cause this race condition.

The handling of this race condition will depend on your Fault Management needs and how alarms are cleared by the agent. One system handled this problem by applying Alarm Reporting alarms as they arrived, buffering them and re-applying them following the completion of Alarm Synchronization. (Alarm Synchronization does have a well-defined completion.) The amount of buffering can be reduced by not buffering alarm conditions that will not cause this race condition.

Other race conditions may potentially exist. Perceived Severity values may be received in different orders. This may or may not be important to a manager.

In addition, if we look at the case where there are no Current Alarms, we may issue an Alarm Synchronization request and receive new alarms before we receive the indication of no Current Alarms.

APPENDIX C – Alarm Synchronization Selection Criteria Example

This will show some examples of the Alarm Synchronization selection criteria. Using Recommendation M.3120 [3] for an example, let's assume the following naming sub-tree:

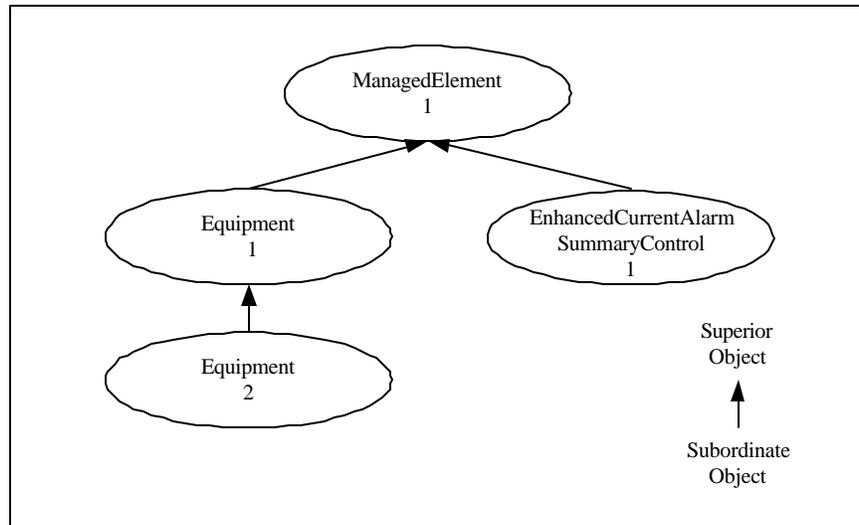


Figure C.1. Sample Naming Tree For Alarm Synchronization Example

Let's also assume the following collection of Current Alarms:

Class	Instance	Current Alarm
Managed Element	1	A
Equipment	1	B
Equipment	2	C

Table C.1. Current Alarm Distribution For Alarm Synchronization Example

In this example, there are three Current Alarms, which we are designating as A, B and C.

Individual examples:

- 1) Alarm Synchronization Info Choice = All Objects Relative To Superior.
Current Alarms retrieved = A, B and C.
- 2) Alarm Synchronization Info Choice = Scoped Criteria.
Base Managed Object = Managed Element 1.
Scope = Whole Sub-tree.
Criteria = "TRUE". (Default filter.)
Current Alarms retrieved = A, B and C. (Effectively the same as the earlier example.)
- 3) Alarm Synchronization Info Choice = Scoped Criteria.
Base Managed Object = Managed Element 1.
Scope = Individual Level.
Level = 1.
Criteria = "TRUE".
Current Alarms retrieved = B.
- 4) Alarm Synchronization Info Choice = Scoped Criteria.
Base Managed Object = Equipment 1.
Scope = Base Object Only.

- Criteria = "TRUE".
Current Alarms retrieved = B.
- 5) Alarm Synchronization Info Choice = Scoped Criteria.
Base Managed Object = Managed Element 1.
Scope = Base To Level.
Level = 1.
Criteria = "TRUE".
Current Alarms retrieved = A, B.
- 6) Alarm Synchronization Info Choice = Object List.
Object Instance #1 = Equipment 1.
Object Instance #2 = Equipment 2.
Current Alarms retrieved = B and C.
- 7) Alarm Synchronization Info Choice = Object List.
Object Instance #1 = Equipment = 1.
Object Instance #2 = Equipment = 2.
Object Instance #3 = Equipment = 3 (i.e., outside the scope).

This will issue an Invalid Parameter error for Object Instance #3.

APPENDIX D – Alarm Surveillance Functions, Services And Functional Units

This appendix updates the Alarm Surveillance section in Recommendation Q.821 [7] according to the capabilities available in CORBA. Appendix D outlines the alarm surveillance functions and services available via Recommendation X.780 [16], Recommendation Q.816 [6] and this document.

The general format of the Recommendation Q.821 Alarm Surveillance section has been maintained. Note, however, that such terms as functional unit do not exist in the CORBA framework [16] as they do in Q / CMISE. In the CORBA framework, functional units are not negotiated as they are with Q / CMISE. Also, the CORBA framework has added new capabilities, such as Heartbeat Service, and new objects, such as Event Channels.

D.1 Alarm Surveillance Functions

Alarm Surveillance functions are used to monitor or interrogate network elements (or both) about events or conditions. Event data is generated by a network element upon the detection of an abnormal condition. Examples of such events are detection of transmission data errors, the violation of a performance threshold and the detection of faulty equipment. Event data can be reported at the time of occurrence, logged for future access or both. An event may also cause further management actions within the network element that lead to the generation of other management data.

The management information related to Alarm Surveillance whose semantics is described includes managed object classes, support object classes and their associated attributes.

The Q / CMISE services from Recommendation Q.821 [7] are shown for comparison purposes only and are not discussed further in this document.

D.1.1 Alarm Reporting Functions

This section describes the Alarm Reporting functions provided by the services specified in this Recommendation when using CORBA. Table D.1 gives a mapping between these functions and the (one or more) services that support each function.

Function	CORBA Service	Q / CMISE Service
Report Alarm	Initiate Event Channel, Terminate Event Channel, Alarm Reporting, Set Event Channel, Get Event Channel, Event Channel Reporting, Heartbeat Reporting, Get Heartbeat Period, Set Heartbeat Period	Alarm Reporting
Route Alarm Report	Initiate Alarm Reporting, Set Event Channel, Obtain Event Channels	Initiate Alarm Reporting, Set Event Forwarding Discriminator
Request Alarm Report Route	Obtain Event Channels	Get Event Forwarding Discriminator
Condition Alarm Reporting	Initiate Alarm Reporting, Terminate Alarm Reporting, Set Event Channel, Obtain Event Channels	Initiate Alarm Reporting, Terminate Alarm Reporting, Set Event Forwarding Discriminator
Request Alarm Report Control Condition	Obtain Event Channels, Get Event Channel	Get Event Forwarding Discriminator
Allow/Inhibit Alarm Reporting	Suspend Alarm Reporting, Resume Alarm Reporting	Suspend Alarm Reporting, Resume Alarm Reporting

Function	CORBA Service	Q / CMISE Service
Request Alarm Report History	Alarm Report Retrieving	Alarm Report Retrieving
Delete Alarm Report History	Alarm Report Deleting	Alarm Report Deleting

Table D.1. Alarm Reporting Functions and CORBA Services

D.1.1.1 Report Alarm

Network element specifies the destination(s) that it will supply a specified set of alarm reports. Network element emits notification upon the occurrence of an alarm.

D.1.1.2 Route Alarm Report

TMN specifies that it will become a consumer of alarm notifications via specified destination(s).

D.1.1.3 Request Alarm Report Route

TMN requests network element to send destination(s) for a specified set of alarm notifications.

D.1.1.4 Condition Alarm Reporting

TMN specifies new attributes for alarm notification destination(s).

D.1.1.5 Request Alarm Report Control Condition

TMN requests to receive current definitions for alarm notification destination(s).

D.1.1.6 Allow/Inhibit Alarm Reporting

TMN requests alarm reporting to be allowed or inhibited to the TMN.

D.1.1.7 Request Alarm Report History

TMN requests and receives specified historical alarm information.

D.1.1.8 Delete Alarm Report History

TMN requests to delete specified historical alarm information.

D.1.2 Alarm Summary Functions

This section describes the Alarm Summary functions provided by the services specified in this Recommendation when using CORBA. Table D.2 gives a mapping between these functions and the (one or more) services that support each function.

Function	CORBA Service	Q / CMISE Service
Report Current Alarm Summary	Current Alarm Summary Reporting	Current Alarm Summary Reporting
Route Current Alarm Summary	<Not Applicable>	Initiate/Set Management Operations Schedule
Request Current Alarm Summary Route	<Not Applicable>	Get Management Operations Schedule
Schedule Current Alarm Summary	Initiate/Terminate/Set Current Alarm Summary Control, Initiate/Terminate/Set Management Operations Schedule	Initiate/Terminate/Set Current Alarm Summary Control, Initiate/Terminate/Set Management Operations Schedule
Request Current Summary Schedule	Get Current Alarm Summary Control, Get Management Operations Schedule	Get Current Alarm Summary Control, Get Management Operations Schedule
Allow/Inhibit Current Alarm Summary	Resume/Suspend Management Operations Schedule	Resume/Suspend Management Operations Schedule
Request Current Alarm Summary	Retrieve Current Alarm Summary	Retrieve Current Alarm Summary

Table D.2. Alarm Summary Functions and CORBA Services

D.1.2.1 Report Current Alarm Summary

Network element provides TMN (based on a pre-defined schedule) with a Current Alarm Summary.

D.1.2.2 Schedule Current Alarm Summary

TMN specifies a schedule for the network element to establish for the reporting of Current Alarm Summaries. The schedule information specifies when it should be reported.

D.1.2.3 Condition Current Alarm Summary

TMN specifies a condition for the network element to establish for the reporting of Current Alarm Summaries. The condition information specifies what should be reported.

D.1.2.4 Request Current Alarm Summary Schedule

TMN requests network element to send the current schedule information for Current Alarm Summary reporting; network element responds with the schedule information.

D.1.2.5 Request Current Alarm Summary Condition

TMN requests network element to send the current condition information for Current Alarm Summary reporting; network element responds with the condition information.

D.1.2.6 Allow/Inhibit Current Alarm Summary

TMN instructs network element to allow/inhibit reporting of the scheduled Current Alarm Summaries.

D.1.2.7 Request Current Alarm Summary

TMN requests the network element to send a Current Alarm Summary; network element responds with the summary.

D.1.3 Alarm Event Criteria Functions

This section describes the Alarm Event Criteria functions provided by the services specified in this Recommendation when using CORBA. Table D.3 gives a mapping between these functions and the (one or more) services that support each function.

Function	CORBA Service	Q / CMISE Service
Condition Alarm Event Criteria	Initiate/Terminate/Set Alarm Severity Assignment Profile	Initiate/Terminate/Set Alarm Severity Assignment Profile
Request Alarm Event Criteria	Get Alarm Severity Assignment Profile	Get Alarm Severity Assignment Profile

Table D.3. Alarm Event Criteria Functions and CORBA Services

D.1.3.1 Condition Alarm Event Criteria

TMN instructs the network element to assign specified alarm attributes (e.g., thresholds, etc.) used by the network element to determine if an event is to be considered an alarm. This function is initially limited to alarm severity assignment.

D.1.3.2 Request Alarm Event Criteria

TMN requests network element to report the current assignments of specified attributes (e.g., thresholds, etc.) used to determine if an event is to be considered an alarm; network element responds with the current assignment of the requested attributes, modes or thresholds. This function is initially limited to the alarm severity attribute.

D.1.4 Alarm Indication Functions

This section describes the Alarm Indication functions provided by the services specified in this Recommendation when using CORBA. Table D.4 gives a mapping between these functions and the (one or more) services that support each function.

Function	CORBA Service	Q / CMISE Service
Inhibit/Allow Audible and Visual Alarm Indications	Set Inhibit/Allow Audible and Visual Local Alarms	Inhibit/Allow Audible and Visual Local Alarms
Request Inhibit/Allow Audible and Visual Local Alarms Indications	Get Inhibit/Allow Audible and Visual Local Alarms	<Not Applicable>
Reset Audible Alarms	Reset Audible Alarms	Reset Audible Alarms

Table D.4. Alarm Indication Functions and CORBA Services

D.1.4.1 Inhibit/Allow Audible and Visual Alarm Indications

TMN instructs the network element to inhibit/allow the operation of specified alarm indication/recording devices such as lamps, speakers, printers, etc.

D.1.4.2 Request Inhibit/Allow Audible and Visual Local Alarms Indication

TMN requests the network element to send the current assignment of the inhibit/allow audible and visual alarm indication; network element responds with the current inhibit/allow audible and visual alarm indication.

D.1.4.3 Reset Audible Alarms

TMN instructs the network element to reset specified audible alarm indicator(s).

D.1.5 Log Control Functions

This section describes the Log Control functions provided by the services specified in this Recommendation when using CORBA. Table D.5 gives a mapping between these functions and the (one or more) services that support each function.

Function	CORBA Service	Q / CMISE Service
Allow / Inhibit Logging	Suspend/Resume Logging	Suspend/Resume Logging
Condition Logging	Initiate/Terminate Log, Set Log	Initiate/Terminate Log, Set Log
Request Log Condition	Get Log, Log Lookup	Get Log

Table D.5. Log Control Functions and CORBA Services

D.1.5.1 Allow / Inhibit Logging

TMN requests that the logging of Log Records be allowed or inhibited.

D.1.5.2 Condition Logging

TMN requests that Log attributes be assigned as specified by the TMN.

D.1.5.3 Request Log Condition

TMN requests the current assignment of specified Log attributes; the TMN receives the current assignment of the specified attributes.

D.2 Alarm Surveillance Service Definition

This section defines the services needed to support the alarm surveillance functions specified in section D.1. Alarm surveillance involves the reporting of alarms and alarm summaries, which are specialized forms of event reporting and the logging of this information.

The services defined to support the alarm surveillance functions specified in section D.1 have been grouped into several functional units to allow negotiation of their use and to allow referencing by other Recommendations. Functional unit negotiation shall be performed as described in [8]. Table D.6 lists these functional units and their corresponding services.

Functional Unit	CORBA Services	Object Classes	Functions
-----------------	----------------	----------------	-----------

Functional Unit	CORBA Services	Object Classes	Functions
Kernel	Initiate Event Channel, Terminate Event Channel, Alarm Reporting, Set Event Channel, Get Event Channel, Event Channel Reporting	Channel Finder, Event Channel Factory, Event Channel, Filter Factory, Filter, Mapping Filter, Sequence Proxy Pull Consumer, Sequence Proxy Pull Supplier, Sequence Proxy Push Consumer, Sequence Proxy Push Supplier, Sequence Pull Consumer, Sequence Pull Supplier, Sequence Push Consumer, Sequence Push Supplier, Structured Proxy Pull Consumer, Structured Proxy Pull Supplier, Structured Proxy Push Consumer, Structured Proxy Push Supplier, Structured Pull Consumer, Structured Pull Supplier, Structured Push Consumer, Structured Push Supplier, Supplier Admin, Typed Proxy Push Consumer, Typed Proxy Push Supplier, Typed Push Consumer	Report Alarm
Basic Alarm Report Control	Suspend Alarm Reporting, Resume Alarm Reporting	Sequence Proxy Pull Consumer, Sequence Proxy Push Supplier, Structured Proxy Pull Consumer, Structured Proxy Push Supplier, Typed Proxy Push Supplier	Allow / Inhibit Alarm Reporting

Functional Unit	CORBA Services	Object Classes	Functions
Enhanced Alarm Report Control	Obtain Event Channel, Initiate Alarm Reporting, Terminate Alarm Reporting	Channel Finder, Consumer Admin, Event Channel, Filter Factory, Filter, Mapping Filter, Sequence Proxy Pull Consumer, Sequence Proxy Pull Supplier, Sequence Proxy Push Consumer, Sequence Proxy Push Supplier, Sequence Pull Consumer, Sequence Pull Supplier, Sequence Push Consumer, Sequence Push Supplier, Structured Proxy Pull Consumer, Structured Proxy Pull Supplier, Structured Proxy Push Consumer, Structured Proxy Push Supplier, Structured Pull Consumer, Structured Pull Supplier, Structured Push Consumer, Structured Push Supplier, Typed Proxy Push Consumer, Typed Proxy Push Supplier, Typed Push Consumer	Condition Alarm Reporting, Route Alarm Report, Request Alarm Report Route, Request Alarm Report Control Condition
Alarm Report Retrieval	Alarm Report Retrieving	Iterator, Notify Log, Typed Notify Log	Request Alarm Report History
Alarm Report Deletion	Alarm Report Deleting	Notify Log, Typed Notify Log	Delete Alarm Report History
Current Alarm Summary Reporting	Current Alarm Summary Reporting	Management Operations Schedule, Current Alarm Summary Control	Report Current Alarm Summary
Basic Management Operations Scheduling	Suspend Management Operations Schedule, Resume Management Operations Schedule	Management Operations Schedule	Allow / Inhibit Current Alarm Summary
Enhanced Management Operations Scheduling	Initiate Management Operations Schedule, Terminate Management Operations Schedule, Set Management Operations Schedule, Get Management Operations Schedule	Management Operations Schedule	Schedule Current Alarm Summary, Route Current Alarm Summary, Request Current Alarm Summary Schedule, Request Current Alarm Summary Route

Functional Unit	CORBA Services	Object Classes	Functions
Current Alarm Summary Reporting Control	Initiate Current Alarm Summary Control, Terminate Current Alarm Summary Control, Set Current Alarm Summary Control, Get Current Alarm Summary Control	Current Alarm Summary Control	Schedule Current Alarm Summary, Request Current Alarm Summary Schedule
Current Alarm Summary Retrieval	Retrieve Current Alarm Summary	Current Alarm Summary Control	Request Current Alarm Summary
Alarm Event Criteria Management	Initiate Alarm Severity Assignment Profile, Terminate Alarm Severity Assignment Profile, Set Alarm Severity Assignment Profile, Get Alarm Severity Assignment Profile	Alarm Severity Assignment Profile	Condition Alarm Event Criteria, Request Alarm Event Criteria
Alarm Indication Management	Set Allow And Inhibit Audible and Visual Local Alarms, Get Allow And Inhibit Audible and Visual Local Alarms, Reset Audible Alarms	Managed Element or its subclasses	Inhibit/Allow Audible and Visual Local Alarm Indications, Request Inhibit/Allow Audible and Visual Local Alarm Indications, Reset Audible Alarm
Basic Log Control	Suspend Logging, Resume Logging	Notify Log	Inhibit/Allow Logging
Enhanced Log Control	Initiate Log, Terminate Log, Set Log, Get Log, Log Lookup	Channel Finder, Event Channel, Filter Factory, Filter, Notify Log Factory, Notify Log, Typed Notify Log Factory, Typed Notify Log	Condition Logging, Request Log Condition
Heartbeat	Heartbeat Reporting, Get Heartbeat Period, Set Heartbeat Period	Heartbeat	Report Alarm

Table D.6. Alarm Surveillance Functional Units, Services, Object Classes and Functions

D.2.1 Kernel Functional Unit

The Kernel functional unit contains the Alarm Reporting service, the Initiate Event Channel, the Terminate Event Channel, the Get Event Channel, the Set Event Channel and the Event Channel Reporting services described below. Figure D.1 shows the interactions between the managing and managed system for this functional unit. Note that the Event Channel shown in Figure D.1 may be predefined.

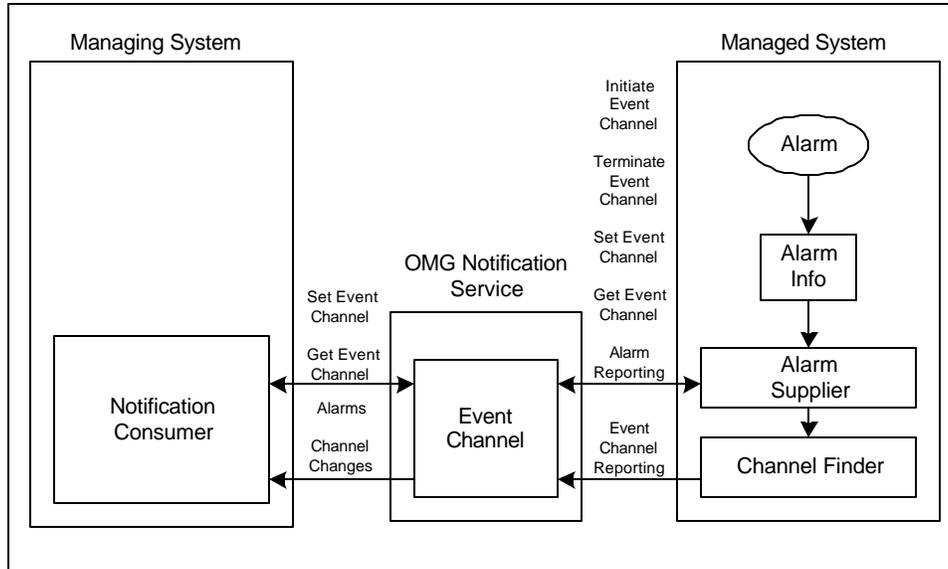


Figure D.1. Kernel Functional Unit

D.2.1.1 Alarm Reporting Service

The Alarm Reporting service allows a managed system to report the detection of an alarm condition for a managed object to its managing system(s). This service supports the Report Alarm function in section D.1.1.

For the service definition, see OMG Notification Service [19], Recommendation X.780 [16] and Recommendation Q.816 [6]. The managed system will either supply a single Structured Event, a sequence of Structured Events or a Typed Event. Which will be used is based on Service Level Agreement.

The Suspect Object List is a parameter to be included in the Additional Information parameter of the Alarm Reporting service. The Suspect Object List parameter identifies objects that may be responsible for an alarm condition. Each listed instance may optionally have a failure responsibility probability associated with it.

D.2.1.2 Initiate Event Channel Service

The Initiate Event Channel service allows a managed system to create an instance of the Event Channel object class. This service supports the Report Alarm functions identified in section D.1.1.

Recommendation Q.816 [6] requires that each Event Channel be registered with the Channel Finder service. Adding, modifying or deleting information from the Channel Finder service will result in a Channel Change notification being sent to each defined Event Channel (see section D.2.1.6).

For the service definition, see OMG Notification Service [19].

D.2.1.3 Terminate Event Channel Service

The Terminate Event Channel service allows a managed system to delete an instance of the Event Channel object class. This service supports the Report Alarm functions identified in section D.1.1.

Recommendation Q.816 [6] requires that each Event Channel be registered with the Channel Finder service. Adding, modifying or deleting information from the Channel Finder service will result in a Channel Change notification being sent to each defined Event Channel (see section D.2.1.6).

For the service definition, see OMG Notification Service [19].

D.2.1.4 Set Event Channel Service

The Set Event Channel service is a service that allows a managed or managing system to alter the criteria used to determine the alarm destinations. This service supports the Report Alarms functions identified in section D.1.1.

Managed systems must only deal with Event Channel supplier information and managing systems must only deal with Event Channel consumer information.

For the service definition, see OMG Notification Service [19].

D.2.1.5 Get Event Channel Service

The Get Event Channel service is a service that allows a managed or managing system to access the criteria used to determine the alarm destinations. This service supports the Report Alarms functions identified in section D.1.1.

Managed systems must only deal with Event Channel supplier information and managing systems must only deal with Event Channel consumer information.

For the service definition, see OMG Notification Service [19].

D.2.1.6 Event Channel Reporting Service

The Event Channel Reporting service allows a managed system to report changes to OMG Notification Service [19] Event Channels.

The Event Channel Reporting service is invoked by the Channel Finder service when an Event Channel is added, deleted or modified. Note that the Channel Finder service starts automatically on system initialization. The Change Channel notification will be sent to all defined Event Channels. This service supports the Report Alarm function identified in section D.1.1.

For the service definition, see Recommendation Q.816 [6].

Table D.7 lists the parameters for the Event Channel Reporting service.

Parameter name	Req/Ind	Rsp/Cnf	Notes
Event Header	M	–	
Fixed Event Header	M	–	
Event Type	M	–	
Domain Name	M	–	“telecommunications”
Type Name	M	–	“itu_x780::Notifications::channelChange”
Event Name	M	–	Null
OptionalHeaderFields	C	–	Only if supplied by application
Filterable Event Body	M	–	
Property Name	M	–	“channelModification”
Property Value	M	–	
Channel Modification	M	–	
Property Name	M	–	“channelInfo”
Property Value	M	–	
Channel Information	M	–	
Channel Id	M	–	
Channel Class	M	–	
Base Objects	M	–	
Name Type	M	–	
Event Types	M	–	May contain one or more Event Type
Scoped Name Type	M	–	
Excluded Event Types	M	–	May contain one or more Extended Event Types
Scoped Name Type	M	–	
Source Classes	M	–	May contain one or more Source Classes

Parameter name	Req/Ind	Rsp/Cnf	Notes
Scoped Name Type	M	–	
Excluded Source Classes	M	–	May contain one or more Excluded Source Classes
Scoped Name Type	M	–	
Channel	M	–	

Table D.7. Event Channel Reporting Service Parameters

The following parameters are defined for use in the Event Channel Reporting service:

Channel Modification

This parameter identifies the type of Event Channel change. It can be either:

- Channel Create
- Channel Delete
- Channel Update

Channel Information

This parameter includes information on the modified Event Channel. These results will include the following parameters:

- Channel Id – A string identifier for the Event Channel.
- Channel Class – Scoped class name of the Event Channel.
- Base Objects – Base managed object instances. Empty list indicates that all managed objects covered by this system are used.
- Event Types – List of Event Types support by this Event Channel. As an example, it could contain “itu_x780::Notifications::equipmentAlarm” if Equipment Alarms are supported (see section 7.2.5.1.1). Empty list indicates all Event Types are used by this Event Channel. Note that the Channel Change notification (i.e., this notification) is not included in this list event though it is supported by all Event Channels.
- Excluded Event Types – If the Event Types field is empty, this can be used to exclude Event Types.
- Source Classes – List of interfaces that send notifications to this Event Channel. Empty list indicates all managed objects covered by the supplied Base Objects are to be used.
- Excluded Source Classes – If the Source Classes field is empty, this can be used to exclude managed objects.
- Channel – Reference to the Event Channel object.

D.2.2 Basic Alarm Report Control Functional Unit

The Basic Alarm Report Control functional unit contains the Suspend Alarm Reporting and the Resume Alarm Reporting services. Figure D.2 shows the interactions between the managing and managed system for this functional unit. Note that the Event Channel object shown in Figure D.2 may be predefined.

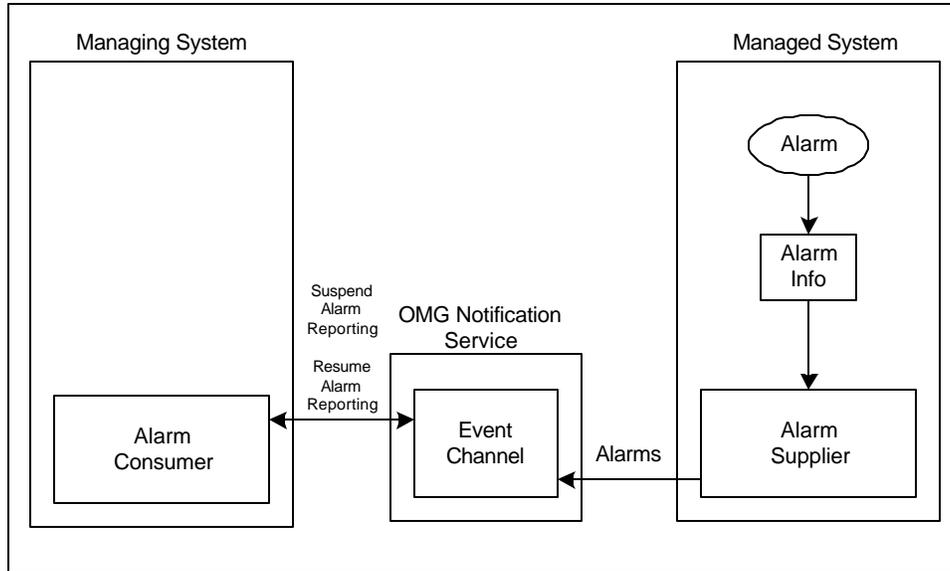


Figure D.2. Basic Alarm Report Control Functional Unit

D.2.2.1 Suspend Alarm Reporting Service

The Suspend Alarm Reporting service allows a managing system to inhibit the reporting of alarm information through an instance of the Sequence Proxy Push Supplier or Structured Proxy Push Supplier object classes. This service supports the Allow / Inhibit Alarm Reporting function identified in section D.1.1.

The managed system will either supply a single Structured Event, a sequence of Structured Events or a Typed Event. Which will be used is based on Service Level Agreement.

Note that managed systems may also inhibit the reporting of alarm information through an instance of the Sequence Proxy Pull Consumer, Sequence Proxy Push Supplier, Structured Proxy Pull Consumer, Structured Proxy Push Supplier, or Typed Proxy Push Supplier object classes.

For the service definition, see OMG Notification Service [19] under Sequence Proxy Pull Consumer, Sequence Proxy Push Supplier, Structured Proxy Pull Consumer, Structured Proxy Push Supplier, or Typed Proxy Push Supplier.

D.2.2.2 Resume Alarm Reporting Service

The Resume Alarm Reporting service allows a managing system to allow the reporting of alarm information through an existing instance of the Sequence Proxy Push Supplier or Structured Proxy Push Supplier object classes. This service supports the Allow / Inhibit Alarm Reporting function identified in section D.1.1.

The managed system will either supply a single Structured Event, a sequence of Structured Events or a Typed Event. Which will be used is based on Service Level Agreement.

Note that managed systems may also allow the reporting of alarm information through an instance of the Sequence Proxy Pull Consumer, Sequence Proxy Push Supplier, Structured Proxy Pull Consumer, Structured Proxy Push Supplier, or Typed Proxy Push Supplier object classes.

For the service definition, see OMG Notification Service [19] under Sequence Proxy Pull Consumer, Sequence Proxy Push Supplier, Structured Proxy Pull Consumer, Structured Proxy Push Supplier, or Typed Proxy Push Supplier.

D.2.3 Enhanced Alarm Report Control Functional Unit

The Enhanced Alarm Report Control functional unit contains the Initiate Alarm Reporting, the Terminate Alarm Reporting and the Obtain Event Channels services. Figure D.3 shows the interactions between the managing and managed system for this functional unit.

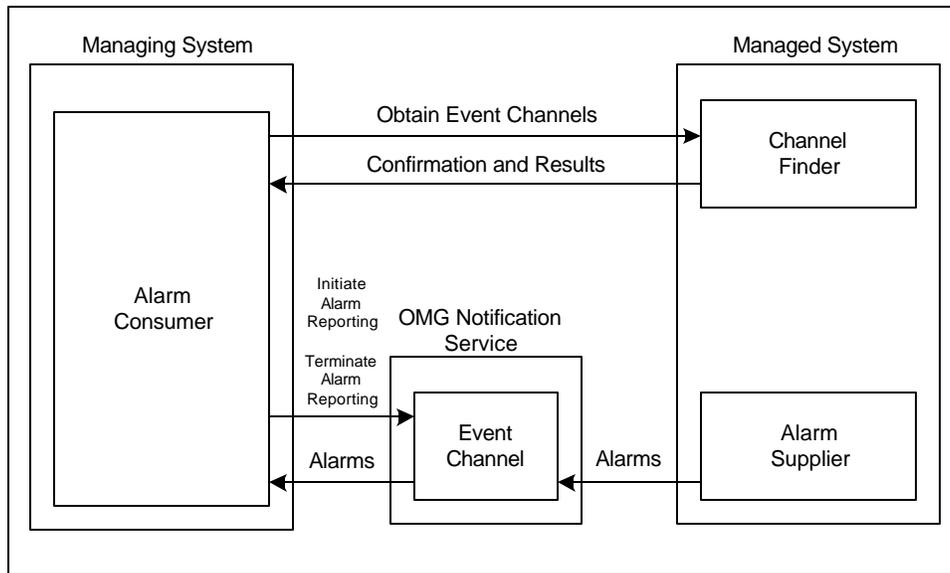


Figure D.3. Enhanced Alarm Report Control Functional Unit

D.2.3.1 Initiate Alarm Reporting Service

The Initiate Alarm Reporting service allows a managing system to connect to an Event Channel. This service supports the Condition Alarm Reporting and the Route Alarm Report functions identified in section D.1.1.

For the service definition, see OMG Notification Service [19].

D.2.3.2 Terminate Alarm Reporting Service

The Terminate Alarm Reporting service allows a managing system to disconnect from an Event Channel. This service supports the Condition Alarm Reporting function identified in section D.1.1.

For the service definition, see OMG Notification Service [19].

D.2.3.3 Obtain Event Channels Service

The Obtain Event Channels service allows a managing system to retrieve what OMG Notification Service [19] Event Channels are defined by the managed system. This service supports the Route Alarm Report, Request Alarm Report Route, Condition Alarm Reporting and Request Alarm Report Control functions identified in section D.1.1.

The semantics of the Channel Finder list method are defined in Recommendation Q.816 [6].

Table D.8 shows the parameters used in the Obtain Event Channels service.

Parameter name	Req/Ind	Rsp/Cnf	Notes
Channel Info Set Type	–	M	May contain zero or more sequences
Channel Id	–	M	
Channel Class	–	M	
Base Objects	–	M	
Name Type	–	M	
Event Types	–	M	May contain one or more Event Type
Scoped Name Type	–	M	
Excluded Event Types	–	M	May contain one or more Extended Event Types

Scoped Name Type	–	M	
Source Classes	–	M	May contain one or more Source Classes
Scoped Name Type	–	M	
Excluded Source Classes	–	M	May contain one or more Excluded Source Classes
Scoped Name Type	–	M	
Channel	–	M	
Exceptions	–	P	

Table D.8. Obtain Event Channels Service

The following parameters are defined for use in the Obtain Event Channels service:

Channel Info Set Type

This parameter includes information on the Event Channels defined by the managed system. These results may include multiple sequences of the following parameters:

- Channel Id – A string identifier for the Event Channel.
- Channel Class – Scoped class name of the Event Channel.
- Base Objects – Base managed object instances. Empty list indicates that all managed objects covered by this system are used.
- Event Types – List of Event Types support by this Event Channel. As an example, it could contain “itu_x780::Notifications::equipmentAlarm” if Equipment Alarms are supported (see section 7.2.5.1.1). Empty list indicates all Event Types are used by this Event Channel. Note that the Channel Change notification is not included in this list event though it is supported by all Event Channels.
- Excluded Event Types – If the Event Types field is empty, this can be used to exclude Event Types.
- Source Classes – List of interfaces that send notifications to this Event Channel. Empty list indicates all managed objects covered by the supplied Base Objects are to be used.
- Excluded Source Classes – If the Source Classes field is empty, this can be used to exclude managed objects.
- Channel – Reference to the Event Channel object.

Exceptions

- Application Error

D.2.4 Alarm Report Retrieval Functional Unit

The Alarm Report Retrieval functional unit contains the Alarm Report Retrieving service described below. Figure D.4 shows the interactions between the managing and managed system for this functional unit.

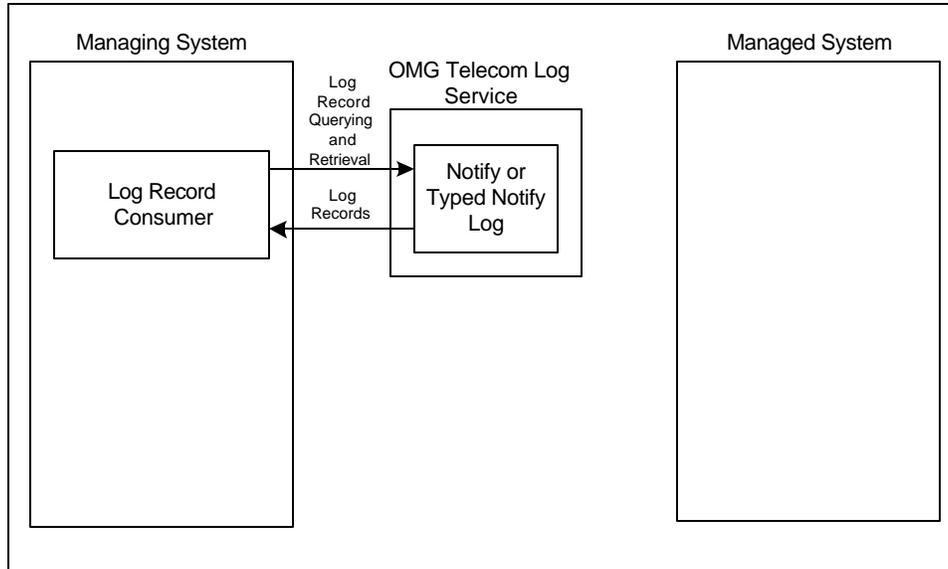


Figure D.4. Alarm Report Retrieval Functional Unit

D.2.4.1 Alarm Report Retrieving Service

The Alarm Report Retrieving service is used to access the values of specified alarm Log Record attributes. This service supports the Request Alarm Report History function identified in section D.1.1.

This service does assume the use of Event Channel filters or Log filters that restrict the log to just alarm notifications (i.e., Communications, Environmental, Equipment, Processing Error or Quality Of Service alarms). Note that OMG Telecom Log Service [20] Log Records do not have sub-classifications (such as Alarm Record) as with Q / CMISE.

This service may be used to retrieve attribute values of a single alarm Log Record by specifying the Log Record Id. In this case, this service utilizes the OMG Telecom Log Service Get Record Attribute method. For the service definition in this case, see the OMG Telecom Log Service.

Alternatively, attributes for multiple alarm Log Records may be retrieved by utilizing the OMG Telecom Log Service Query or Retrieve methods. The Query method returns all alarm Log Records that match a supplied filter. The Retrieve method returns all alarm Log Records starting from a given time. For the service definition in these cases, see the OMG Telecom Log Service.

D.2.5 Alarm Report Deletion Functional Unit

This functional unit contains the Alarm Report Deleting service. Figure D.5 shows the interaction between the managing and managed systems for this functional unit.

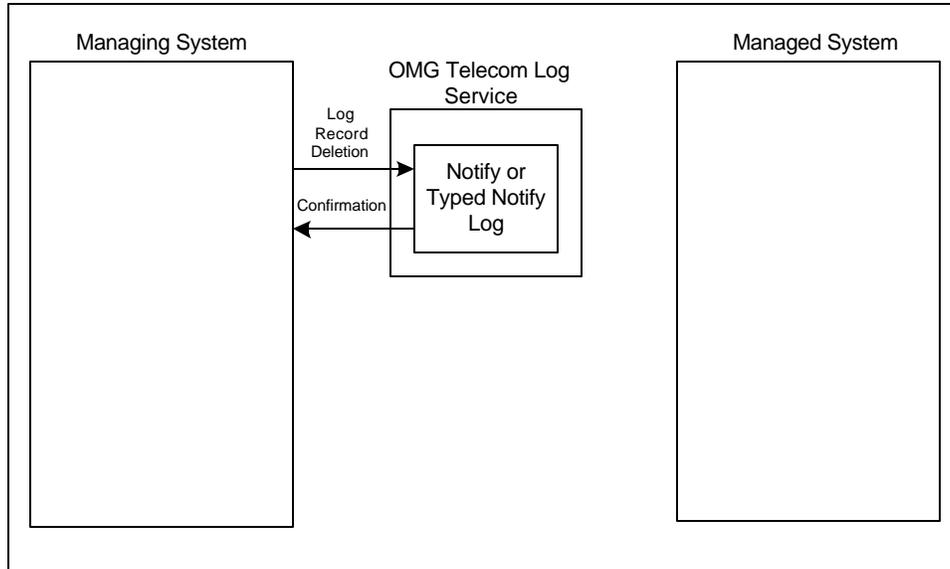


Figure D.5. Alarm Report Deletion Functional Unit

D.2.5.1 Alarm Report Deleting Service

The Alarm Report Deleting service is used to remove specific alarm Log Records. This service supports the Delete Alarm Report History function described in section D.1.1.

For the service definition, see OMG Telecom Log Service [20] under Log Record deletion.

D.2.6 Current Alarm Summary Reporting Functional Unit

The Current Alarm Summary Reporting functional unit contains the Current Alarm Summary Reporting service described below. Figure D.6 shows the interactions between the managing and managed system for this functional unit. Note that the Management Operations Schedule and Current Alarm Summary Control objects shown in Figure D.6 may be predefined. The Management Operations Schedule object shall be present but need not be modifiable by the managing system.

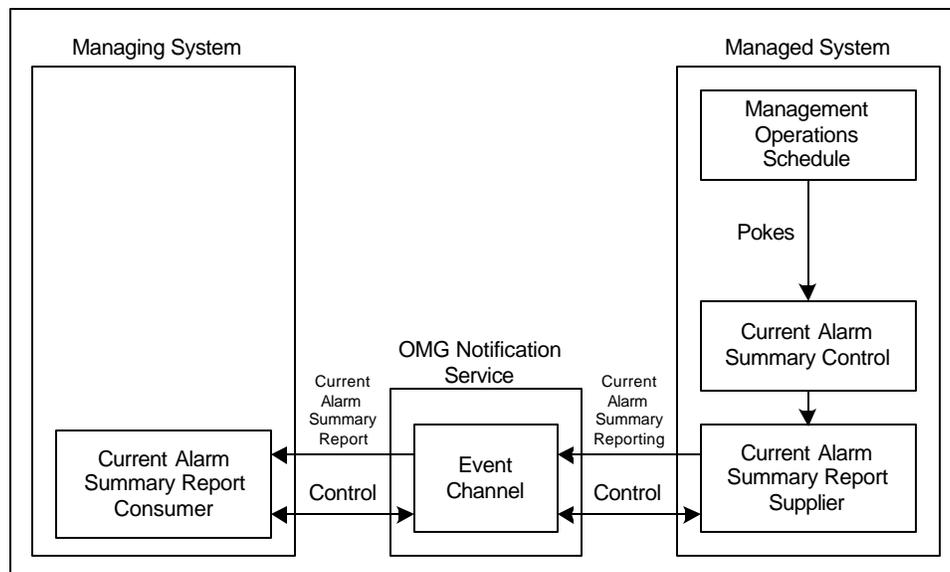


Figure D.6. Current Alarm Summary Reporting Functional Unit

D.2.6.1 Current Alarm Summary Reporting Service

The Current Alarm Summary Reporting service allows a managed system to report a summary of the alarm conditions of specified managed objects to its managing system(s).

The Current Alarm Summary Reporting service is invoked when the Current Alarm Summary Control object pointed to by the Management Operations Schedule object (via the Affected Object Class and Affected Object Instance attributes) is poked. This service supports the Report Current Alarm Summary function identified in section D.1.2.

Table D.9 lists the parameters for the Current Alarm Summary Reporting service.

Parameter name	Req/Ind	Rsp/Cnf	Notes
Event Header	M	–	
Fixed Event Header	M	–	
Event Type	M	–	
Domain Name	M	–	“telecommunications”
Type Name	M	–	“itu_x780::Notifications::current AlarmSummaryReport”
Event Name	M	–	Null
OptionalHeaderFields	C	–	Only if supplied by application
Filterable Event Body	M	–	
Property Name	M	–	“alarmSummaryData”
Property Value	M	–	
Alarm Summary Data	M	–	Zero or more sequences of ObjectAlarmSummaryType
Object Of Reference	C	–	
Summary Info	C	–	
Perceived Severity	C	–	
Alarm Status	C	–	
Probable Cause	C	–	

Table D.9. Current Alarm Summary Reporting Service Parameters

The following parameters are defined for use in the Current Alarm Summary Reporting service:

Alarm Summary Data

This parameter includes the results of an alarm summary report generation by a managed system. These results potentially include multiple sequences of the following parameters:

- Object Of Reference
- Summary Info. This includes multiple sequences of the following parameters (in each sequence, at least one must be provided):
 - Perceived Severity [6] (Optional)
 - Alarm Status [6] (Optional)
 - Probable Cause [6] (Optional)

D.2.7 Basic Management Operations Scheduling Functional Unit

The Basic Management Operations Scheduling functional unit contains the Suspend Management Operations Schedule and the Resume Management Operations Schedule services. Figure D.7 shows the interactions between the managing and managed system for this functional unit. Note that the Management Operations Schedule object shown in Figure D.7 may be predefined. In such cases, only the Administrative State attribute is modifiable by the Managing System.

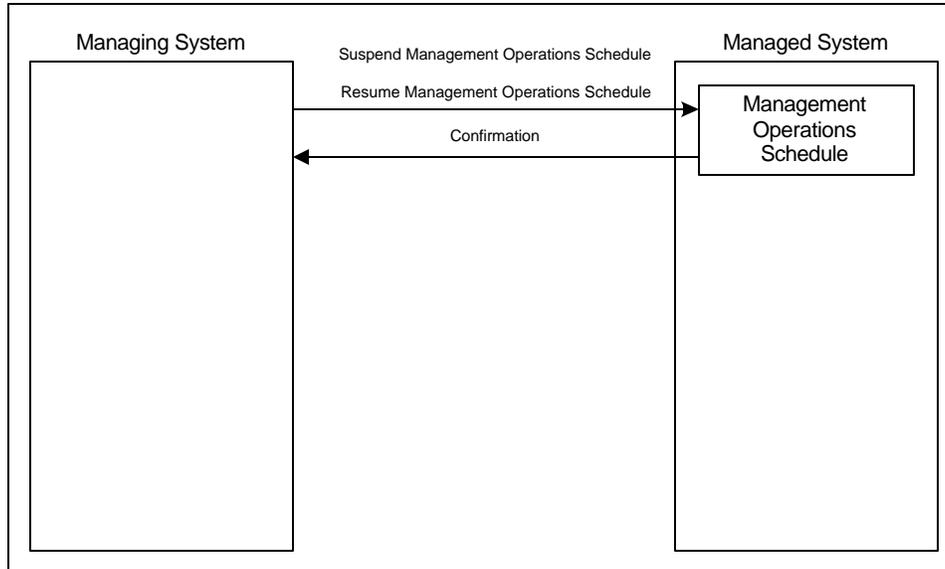


Figure D.7. Basic Management Operations Scheduling Functional Unit

D.2.7.1 Suspend Management Operations Schedule Service

The Suspend Management Operations Schedule service allows a managing system to inhibit the scheduled operation of a service (such as the Current Alarm Summary Reporting service) triggered by an instance of the Management Operations Schedule object class in a managed system. This service utilizes the Set service and procedures defined in [16]. This service supports the Allow / Inhibit Current Alarm Summary function identified in section D.1.2.

The semantics of the Management Operations Schedule attributes are defined in section 6.2.2.

D.2.7.2 Resume Management Operations Schedule Service

The Resume Management Operations Schedule service allows a managing system to resume the scheduled operation of a service (such as the Current Alarm Summary Reporting service) triggered by an instance of the Management Operations Schedule object class in a managed system. This service utilizes the Set service and procedures defined in [16]. This service supports the Allow / Inhibit Current Alarm Summary function identified in section D.1.2.

The semantics of the Management Operations Schedule attributes are defined in section 6.2.2.

D.2.8 Enhanced Management Operations Scheduling Functional Unit

The Enhanced Management Operations Scheduling functional unit contains the Initiate Management Operations Schedule service, the Terminate Management Operations Schedule, the Set Management Operations Schedule and the Get Management Operations Schedule services. Figure D.8 shows the interactions between the managing and managed system for this functional unit.

If a bilateral agreement exists between two error reporting service users, the Initiate and Terminate Management Operations Schedule services can be omitted. In this case, operation of the Management Operations Schedule starts automatically at system initialization.

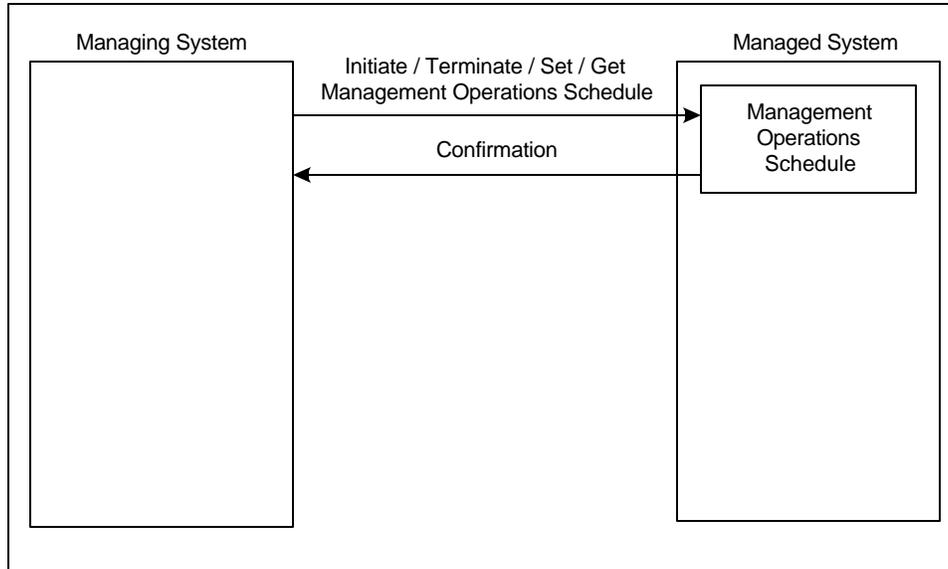


Figure D.8. Enhanced Management Operations Scheduling Functional Unit

D.2.8.1 Initiate Management Operations Schedule Service

The Initiate Management Operations Schedule service allows a managing system to create an instance of the Management Operations Schedule object class in a managed system. This service utilizes the Create service and procedures defined in [16]. This service supports the Schedule Current Alarm Summary and Route Current Alarm Summary functions identified in section D.1.2.

The semantics of the Management Operations Schedule attributes are defined in section 6.2.2.

D.2.8.2 Terminate Management Operations Schedule Service

The Terminate Management Operations Schedule service allows a managing system to delete an instance of the Management Operations Schedule object class in a managed system. This service utilizes the Delete service and procedures defined in [16]. This service supports the Schedule Current Alarm Summary function identified in section D.1.2.

The semantics of the Management Operations Schedule attributes are defined in section 6.2.2.

D.2.8.3 Set Management Operations Schedule Service

The Set Management Operations Schedule service is a confirmed service that allows a managing system to set the attribute values of a specified instance of a Management Operations Schedule object. This service utilizes the Set service and procedures defined in [16]. This service supports the Schedule Current Alarm Summary and Route Current Alarm Summary functions identified in section D.1.2.

The semantics of the Management Operations Schedule attributes are defined in section 6.2.2.

D.2.8.4 Get Management Operations Schedule Service

The Get Management Operations Schedule service allows a managing system to retrieve the values of given attributes of a specified instance of a Management Operations Schedule object. This service utilizes the Get service and procedures defined in [16]. This service supports the Request Current Alarm Summary Schedule and Request Current Alarm Summary Route functions identified in section D.1.2.

The semantics of the Management Operations Schedule attributes are defined in section 6.2.2.

D.2.9 Current Alarm Summary Reporting Control Functional Unit

The Current Alarm Summary Reporting Control functional unit contains the Initiate Current Alarm Summary Control, the Terminate Current Alarm Summary Control, the Set Current Alarm Summary Control and the Get Current Alarm

Summary Control services. Figure D.9 shows the interactions between the managing and managed system for this functional unit.

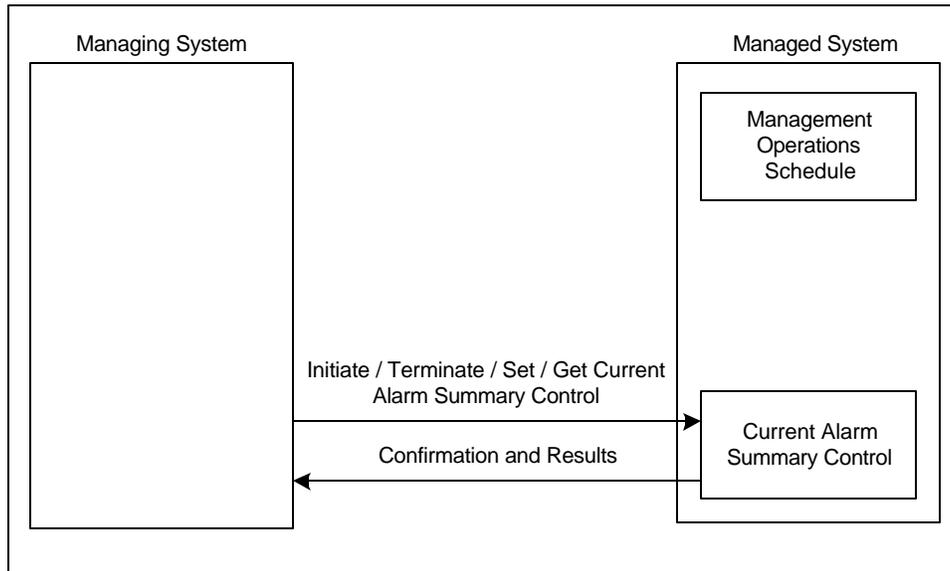


Figure D.9. Current Alarm Summary Reporting Control Functional Unit

D.2.9.1 Initiate Current Alarm Summary Control Service

The Initiate Current Alarm Summary Control service allows a managing system to create an instance of the Current Alarm Summary Control object class in a managed system. This service utilizes the Create service and procedures defined in [16]. This service supports the Schedule Current Alarm Summary function identified in section D.1.2.

The semantics of the Current Alarm Summary Control attributes are defined in section 6.2.1.

D.2.9.2 Terminate Current Alarm Summary Control Service

The Terminate Current Alarm Summary Control service allows a managing system to delete an instance of the Current Alarm Summary Control object class in a managed system. This service utilizes the Delete service and procedures defined in [16]. This service supports the Schedule Current Alarm Summary function identified in section D.1.2.

The semantics of the Current Alarm Summary Control attributes are defined in section 6.2.1.

D.2.9.3 Set Current Alarm Summary Control Service

The Set Current Alarm Summary Control service is a confirmed service that allows a managing system to set the attribute values of a specified instance of a Current Alarm Summary Control object. This service utilizes the Set service and procedures defined in [16]. This service allows a managing system to alter the criteria used to select objects to be included in Current Alarm Summary reports. This service supports the Schedule Current Alarm Summary function identified in section D.1.2.

The semantics of the Current Alarm Summary Control attributes are defined in section 6.2.1.

D.2.9.4 Get Current Alarm Summary Control Service

The Get Current Alarm Summary Control service allows a managing system to retrieve the values of given attributes of a specified instance of a Current Alarm Summary Control object. This service utilizes the Get service and procedures defined in [16]. This service supports the Request Current Alarm Summary Schedule function identified in section D.1.2.

The semantics of the Current Alarm Summary Control attributes are defined in section 6.2.1.

D.2.10 Current Alarm Summary Retrieval Functional Unit

The Current Alarm Summary Retrieval functional unit contains the Retrieve Current Alarm Summary service described below. Figure D.10 shows the interactions between the managing and managed system for this functional unit. Note that the Current Alarm Summary Control object shown in Figure D.10 may be predefined. If this is the only Current Alarm Summary-related functional unit supported, the Current Alarm Summary Control object class shall be present but need not be modifiable by the managing system.

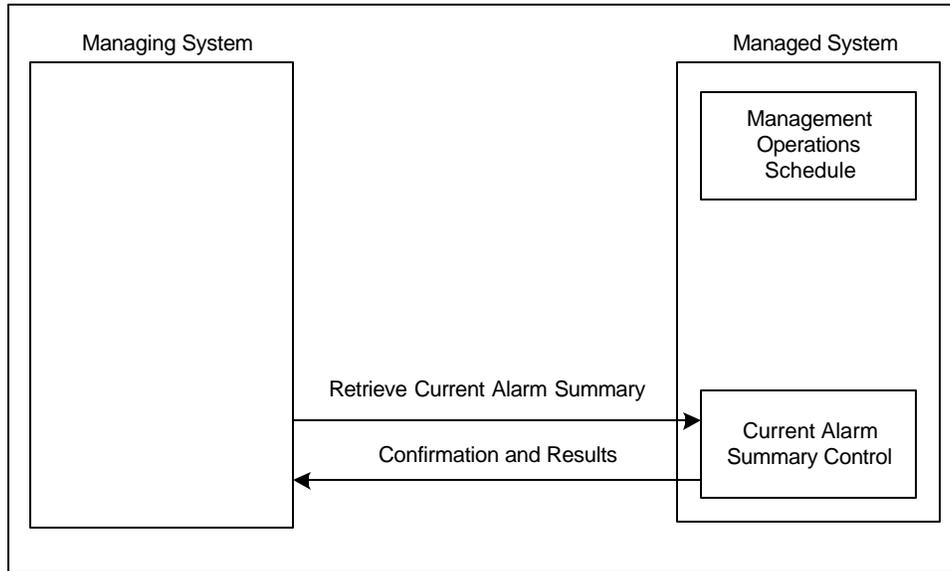


Figure D.10. Current Alarm Summary Retrieval Functional Unit

D.2.10.1 Retrieve Current Alarm Summary Service

The Retrieve Current Alarm Summary service is used to request that a Current Alarm Summary report be sent from the managed system to the managing system. It utilizes the retrieveCurrentAlarmSummary method of the Current Alarm Summary object. It supports the Request Current Alarm Summary function identified in section D.1.2.

Table D.10 shows the parameters used in the Retrieve Current Alarm Summary service.

Parameter name	Req/Ind	Rsp/Cnf	Notes
Summary Contents	U	–	One, two or all three may be included
Include Perceived Severity	C	–	
Include Alarm Status	C	–	
Include Probable Cause	C	–	
Alarm Summary Data	–	M	May be NULL if no alarm summary report
Object Of Reference	–	C	
Summary Info	–	C	
Perceived Severity	–	C	Depending whether Include Perceived Severity used
Alarm Status	–	C	Depending whether Include Alarm Status used
Probable Cause	–	C	Depending whether Include Alarm Status used
Exceptions	–	P	

Table D.10. Retrieve Current Alarm Summary Service Parameters

The following parameters are defined for use in the Retrieve Current Alarm Summary service:

Summary Contents

This parameter is used to control the attributes that shall be included in the report. The report may include any (and multiples) of the following:

- Perceived Severity [6]
- Alarm Status [6]
- Probable Cause [6]

Alarm Summary Data

This parameter includes the results of an alarm summary report generation by a managed system. These results include multiple sequences of the following parameters:

- Object Of Reference
- Summary Info. This includes multiple sequences of the following parameters:
 - Perceived Severity [6] (Optional)
 - Alarm Status [6] (Optional)
 - Probable Cause [6] (Optional)

Exceptions

- Application Error

D.2.11 Alarm Event Criteria Management Functional Unit

The Alarm Event Criteria Management functional unit contains the Set Alarm Severity Assignment List and Get Alarm Severity Assignment List services. Figure D.11 shows the interactions between the managing and managed system for this functional unit.

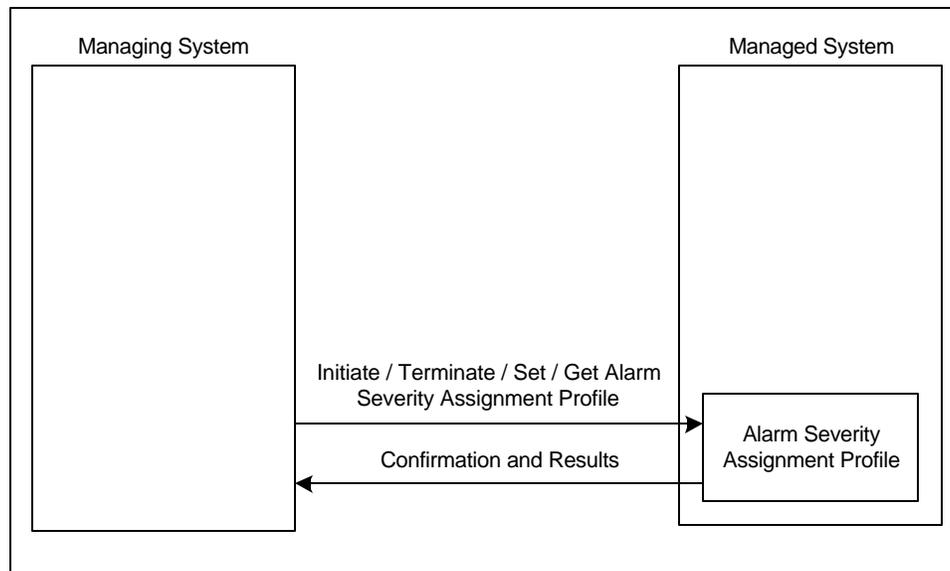


Figure D.11. Alarm Event Criteria Management Functional Unit

D.2.11.1 Initiate Alarm Severity Assignment Profile Service

The Initiate Alarm Severity Assignment Profile service allows a managing system to create an instance of the Alarm Severity Assignment Profile object class in a managed system. This service utilizes the Create service and procedures defined in [16]. This service supports the Condition Alarm Event Criteria function identified in section D.1.3.

The semantics of the Alarm Severity Assignment Profile attributes are defined in [3].

D.2.11.2 Terminate Alarm Severity Assignment Profile Service

The Terminate Alarm Severity Assignment Profile service allows a managing system to delete an instance of the Alarm Severity Assignment Profile object class in a managed system. This service utilizes the Delete service and procedures defined in [16]. This service supports the Condition Alarm Event Criteria function identified in section D.1.3.

The semantics of the Alarm Severity Assignment Profile attributes are defined in [3].

D.2.11.3 Set Alarm Severity Assignment Profile Service

The Set Alarm Severity Assignment Profile service allows a managing system to modify the alarm severity assignment list associated with the Alarm Severity Assignment Profile object instance. This service utilizes the Set service and procedures defined in [16]. This service supports the Condition Alarm Event Criteria function identified in section D.1.3.

The semantics of the Alarm Severity Assignment Profile object class are described in [3].

D.2.11.4 Get Alarm Severity Assignment Profile Service

The Get Alarm Severity Assignment Profile service allows a managing system to retrieve the alarm severity assignment list associated with the Alarm Severity Assignment Profile object instance. This service utilizes the Get service and procedures defined in [16]. This service supports the Request Alarm Event Criteria function identified in section D.1.3.

The semantics of the Alarm Severity Assignment Profile object class are described in [3].

D.2.12 Alarm Indication Management Functional Unit

The Alarm Indication Management functional unit contains the Set Allow And Inhibit Audible and Visual Local Alarms, the Get Allow And Inhibit Audible and Visual Local Alarms and the Reset Audible Alarm services. Figure D.12 shows the interactions between the managing and managed system for this functional unit.

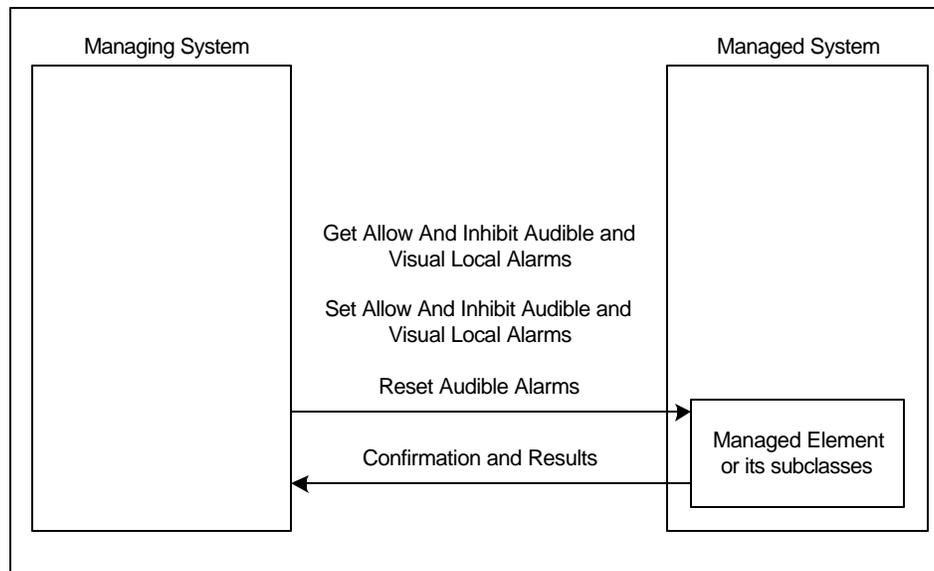


Figure D.12. Alarm Indication Management Functional Unit

D.2.12.1 Set Allow And Inhibit Audible and Visual Local Alarms Service

The Set Allow and Inhibit Audible and Visual Local Alarms service allows a managing system to allow or inhibit audible and visual local alarms. This service utilizes the enableAudibleVisualLocalAlarmSet method [3]. This service supports the Inhibit/Allow Audible and Visual Local Alarm Indications function identified in section D.1.4.

Table D.11 shows the parameters used in the Set Allow and Inhibit Audible and Visual Local Alarms service.

Parameter name	Req/Ind	Rsp/Cnf	Notes
Enable	M	–	
Exceptions	–	P	

Table D.11. Set Allow and Inhibit Audible and Visual Local Alarms Service

The following parameters are defined for use in the Set Allow and Inhibit Audible and Visual Local Alarms service:

Enable

If set to TRUE, allow audible and visual local alarms. If set to FALSE, inhibit audible and visual local alarms.

Exceptions

- Application Error
- No Audible Visual Local Alarm Package

An Attribute Value Change event notification will be issued under the following conditions:

- The Allow and Inhibit Audible and Visual Local Alarms changed value from TRUE to FALSE or from FALSE to TRUE,
- The Attribute Value Change event notification is supported for this managed object instance and
- The Allow and Inhibit Audible and Visual Local Alarms attribute is a monitored attribute for the Attribute Value Change event notification.

D.2.12.2 Get Allow And Inhibit Audible and Visual Local Alarms Service

The Get Allow and Inhibit Audible and Visual Local Alarms service allows a managing system to query whether the audible and visual local alarms are allowed or inhibited. This service utilizes the enableAudibleVisualLocalAlarmGet method [3]. This service supports the Request Inhibit/Allow Audible and Visual Local Alarm Indications function identified in section D.1.4.

Table D.12 shows the parameters used in the Get Allow and Inhibit Audible and Visual Local Alarms service.

Parameter name	Req/Ind	Rsp/Cnf	Notes
Results	–	M	
Exceptions	–	P	

Table D.12. Get Allow and Inhibit Audible and Visual Local Alarms Service

The following parameters are defined for use in the Get Allow and Inhibit Audible and Visual Local Alarms service:

Results

If set to TRUE, audible and visual local alarms are allowed. If set to FALSE, audible and visual local alarms are inhibited.

Exceptions

- Application Error
- No Audible Visual Local Alarm Package

D.2.12.3 Reset Audible Alarms Service

The Reset Audible Alarms service allows a managing system to retire existing audible and visual local alarms without inhibiting them in the future. This service utilizes the resetAudibleAlarm method [3]. This service supports the Reset Audible Alarms function identified in section D.1.4.

Table D.13 shows the parameters used in the Reset Audible Alarm service.

Parameter name	Req/Ind	Rsp/Cnf	Notes
----------------	---------	---------	-------

Exceptions	-	P	
------------	---	---	--

Table D.13. Reset Audible Alarms Service

The following parameters are defined for use in the Reset Audible Alarms service:

Exceptions

- Application Error
- No Audible Visual Local Alarm Package

D.2.13 Basic Log Control Functional Unit

The Basic Log Control functional unit contains the Suspend Logging and the Resume Logging services. Figure D.13 shows the interactions between the managing and managed system for this functional unit. Note that the Log object shown in Figure D.13 may either have a predefined Filter or the Filter may be absent.

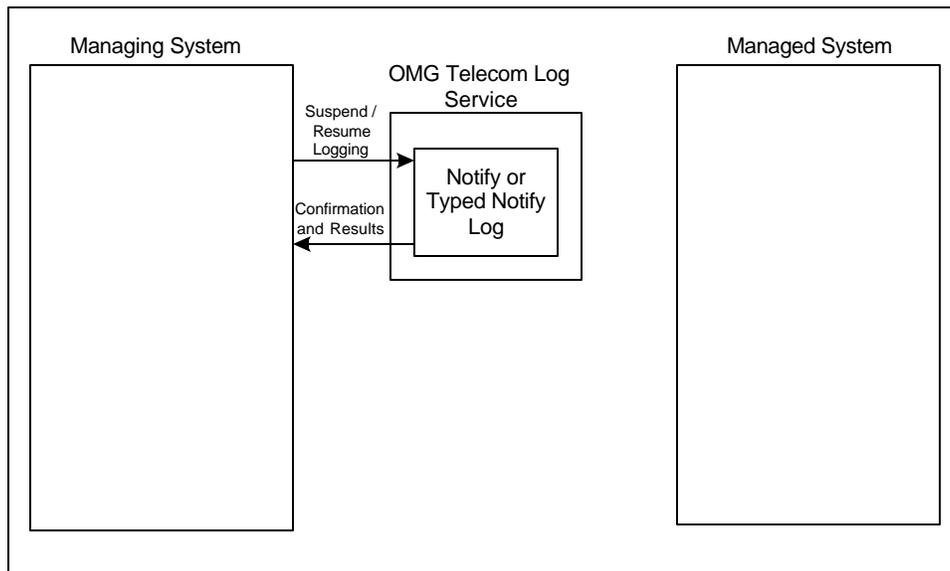


Figure D.13. Basic Log Control Functional Unit

D.2.13.1 Suspend Logging Service

The Suspend Logging service allows a managing system to inhibit the logging of Log Records. This service supports the Inhibit / Allow Logging function identified in section D.1.5.

For the service definition, see OMG Telecom Log Service [20] and the use of Operational State.

D.2.13.2 Resume Logging Service

The Resume Logging service allows a managing system to resume the logging of Log Records. This service supports the Inhibit / Allow Logging function identified in section D.1.5.

For the service definition, see OMG Telecom Log Service [20] and the use of Operational State.

D.2.14 Enhanced Log Control Functional Unit

The Enhanced Log Control functional unit contains the Initiate Log, the Terminate Log, the Set Log, the Get Log and the Log Lookup services. Figure D.14 shows the interactions between the managing and managed system for this functional unit. Note that the Log object shown in Figure D.14 may either be predefined or be created using the Initiate Log service.

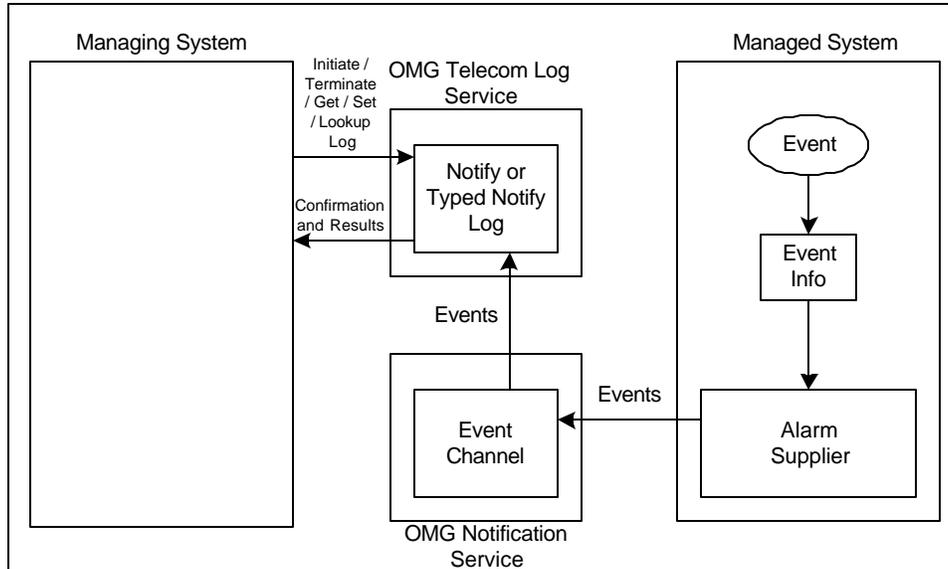


Figure D.14. Enhanced Log Control Functional Unit

D.2.14.1 Initiate Log Service

The Initiate Log service allows a managing system to create an instance of the Log object class in the OMG Telecom Log Service [20]. This service supports the Condition Logging function identified in section D.1.5.

Log Event Channels must be registered with the Channel Finder service (see Recommendation Q.816 [6]).

Notifications of type Structured Event, Event Batch or Typed Event may be stored in a Notify Log. Notifications of type Typed Event may be stored in a Typed Notify Log.

Logs and Log factories may generate notifications that can be accessed via the Route Alarm Report function. Note that OMG Telecom Log Service notifications may be of a different format from notifications generated from Recommendation Q.816 [6]. The following types of notifications may be generated:

- a) Object creation
- b) Object deletion
- c) Threshold alarm
- d) Attribute value change
- e) State change
- f) Processing error alarm

For the service definition, see OMG Telecom Log Service under NotifyLogFactory or TypedNotifyLogFactory.

D.2.14.2 Terminate Log Service

The Terminate Log service allows a managing system to delete an instance of the Log object class in the OMG Telecom Log Service [20]. This service supports the Condition Logging function identified in section D.1.5.

For the service definition, see OMG Telecom Log Service under NotifyLogFactory or TypedNotifyLogFactory.

D.2.14.3 Set Log Service

The Set Log service is a confirmed service that allows a managing system to set the attribute values of a specified instance of a Log object. This service supports the Condition Logging function identified in section D.1.5.

The Set Log service allows the setting of:

- a) Administrative state
- b) Availability status
- c) Maximum log size
- d) Log full action
- e) Log duration
- f) Log scheduling
- g) Log capacity thresholds
- h) Expiration time for Log Records
- i) Quality Of Service (QOS) properties
- j) Log filter

For the service definition, see OMG Telecom Log Service [20] under NotifyLog or TypedNotifyLog.

D.2.14.4 Get Log Service

The Get Log service allows a managing system to retrieve the values of given attributes of a specified instance of a Log object. This service supports the Request Log Condition function identified in section D.1.5.

For the service definition, see OMG Telecom Log Service [20] under NotifyLog or TypedNotifyLog.

D.2.14.5 Log Lookup Service

The Log Lookup service is a confirmed service that allows a managing system to query which Log objects are in existence. This service supports the Condition Logging function identified in section D.1.5.

For the service definition, see OMG Telecom Log Service [20] under NotifyLogFactory.

D.2.15 Heartbeat Functional Unit

The Heartbeat functional unit contains the Heartbeat Reporting, the Get Heartbeat Period and the Set Heartbeat Period services. Figure D.15 shows the interactions between the managing and managed system for this functional unit. Note that the Event Channel shown in Figure D.15 may be predefined.

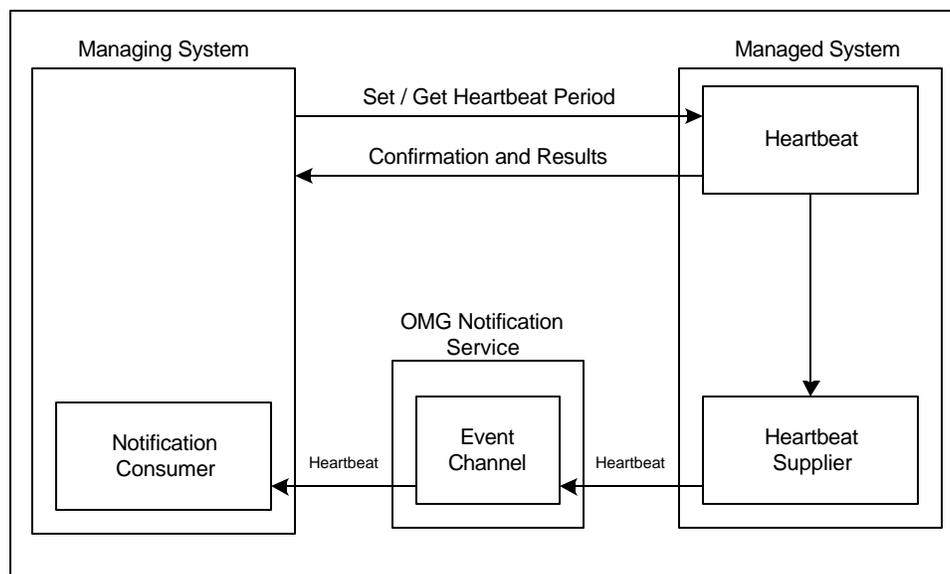


Figure D.15. Heartbeat Functional Unit

D.2.15.1 Heartbeat Reporting Service

The Heartbeat Reporting service allows a managed system to send a periodic heartbeat notification. When active, a heartbeat notification is sent to each registered Event Channel. This service supports the Report Alarms functions identified in section D.1.1. For the service definition, see Recommendation Q.816 [6].

Table D.14 lists the parameters for the Heartbeat Reporting service.

Parameter name	Req/Ind	Rsp/Cnf	Notes
Event Header	M	–	
Fixed Event Header	M	–	
Event Type	M	–	
Domain Name	M	–	“telecommunications”
Type Name	M	–	“itu_x780::Notifications::Heartbeat”
Event Name	M	–	Null
OptionalHeaderFields	C	–	Only if supplied by application
Filterable Event Body	M	–	
Property Name	M	–	“systemLabel”
Property Value	M	–	
System Label	M		
Property Name	M	–	“channelID”
Property Value	M	–	
Channel Id	M		
Property Name	M	–	“period”
Property Value	M	–	
Period	M		
Property Name	M	–	“timeStamp”
Property Value	M	–	
Time Stamp	M		

Table D.14. Heartbeat Reporting Service Parameters

The following parameters are defined for use in the Heartbeat Reporting service:

- System Label – String identification of the managed system.
- Channel Id – String Event Channel id.
- Period – Value of the heartbeat period (in seconds).
- Time Stamp – Time when heartbeat notification is emitted.

D.2.15.2 Set Heartbeat Period Service

The Set Heartbeat Period service allows a managing system to set the heartbeat period. This service utilizes the periodSet method [6]. This service supports the Report Alarms functions identified in section D.1.1.

Table D.15 shows the parameters used in the Set Heartbeat Period service.

Parameter name	Req/Ind	Rsp/Cnf	Notes
Period	M	–	
Exceptions	–	P	

Table D.15. Set Heartbeat Period Service

The following parameters are defined for use in the Set Heartbeat Period service:

Period

The interval, in seconds, at which the Heartbeat Reporting service emits a heartbeat notification. Setting the period results in the issuing of a heartbeat notification. If it is set to zero, no further heartbeat notifications will be issued.

Exceptions

- Application Error
- Invalid Parameter

D.2.15.3 Get Heartbeat Period Service

The Get Heartbeat Period service allows a managing system to access the heartbeat period. This service utilizes the periodGet method [6]. This service supports the Report Alarms functions identified in section D.1.1.

Table D.16 shows the parameters used in the Get Heartbeat Period service.

Parameter name	Req/Ind	Rsp/Cnf	Notes
Period	–	M	
Exceptions	–	P	

Table D.16. Get Heartbeat Period Service

The following parameters are defined for use in the Get Heartbeat Period service:

Period

The interval, in seconds, at which the Heartbeat Reporting service emits a heartbeat notification. If it is zero, no heartbeat notifications will be emitted.

Exceptions

- Application Error

D.3 Protocol Specification

D.3.1 Elements Of Procedure

Except for the services identified below, this specification makes use of the elements of procedure defined for the services described in section D.2.

D.3.1.1 Managed Objects

This specification references the following support objects whose IDL is specified in OMG Notification Service [19]:

- a) Channel Finder
- b) Consumer Admin
- c) Event Channel
- d) Event Channel Factory
- e) Filter
- f) Filter Factory
- g) Mapping Filter
- h) Sequence Proxy Pull Consumer
- i) Sequence Proxy Pull Supplier
- j) Sequence Proxy Push Consumer
- k) Sequence Proxy Push Supplier
- l) Sequence Pull Consumer
- m) Sequence Pull Supplier
- n) Sequence Push Consumer
- o) Sequence Push Supplier
- p) Structured Proxy Pull Consumer
- q) Structured Proxy Pull Supplier
- r) Structured Proxy Push Consumer
- s) Structured Proxy Push Supplier
- t) Structured Pull Consumer
- u) Structured Pull Supplier
- v) Structured Push Consumer
- w) Structured Push Supplier
- x) Supplier Admin
- y) Typed Proxy Push Consumer

- z) Typed Proxy Push Supplier
- aa) Typed Push Consumer

This specification references the following support objects whose IDL is specified in OMG Telecom Log Service [20]:

- a) Iterator
- b) Notify Log
- c) Notify Log Factory
- d) Typed Notify Log
- e) Typed Notify Log Factory

This specification references the following managed objects whose IDL are specified in Recommendation M.3120 [3]:

- a) Alarm Severity Assignment Profile
- b) Managed Element
- c) Managed Element Complex
- d) Network

This specification references the following managed objects whose IDL are specified in Recommendation X.780 [16]:

- a) Managed Object
- b) Managed Object factory

This specification references the following support objects whose IDL are specified in this document (see section 10):

- a) Alarm Synchronization Data Iterator
- b) Current Alarm Summary Control
- c) Current Alarm Summary Control Factory
- d) Enhanced Current Alarm Summary Control
- e) Enhanced Current Alarm Summary Control Factory
- f) Management Operations Schedule
- g) Management Operations Schedule Factory

D.3.1.2 Valuetypes

This specification references the following valuetypes specified in this document whose IDL are defined in Recommendation X.780 [16]:

- a) Managed Object Value Type

D.3.1.3 Types

This specification references the following types associated with the objects specified in this document whose IDL are defined in Recommendation X.780 [16]:

- a) AdministrativeStateType
- b) BitStringType
- c) DeletePolicyType
- d) ExternalTimeType
- e) GeneralizedTimeType
- f) NameType
- g) NameBindingType
- h) ObjectClassType
- i) OperationalStateType
- j) PerceivedSeverityType
- k) ProbableCauseType

- l) StartTimeType
- m) StopTimeType
- n) StringSetType

This specification references the following types associated with the objects specified in this document whose IDL are defined in Recommendation Q.816 [6]:

- a) FilterType
- b) LanguageType
- c) ScopeType

This specification references the following types associated with the objects specified in section 6 whose IDL is defined in Recommendation M.3120 [3]:

- a) AlarmStatusType

This specification references the following types associated with the objects specified in section 6 whose IDL is defined in OMG Notification Service [19]:

- a) EventBatch

The objects defined in this specification inherit attributes from ManagedObject as specified in Recommendation X.780 [16]; these attributes are not repeated here.

This specification references the following types that are defined in section 10.3:

- a) AffectedObjectClassType
- b) AffectedObjectInstancesType
- c) AlarmStatusSetType
- d) AlarmStatusTypeOpt
- e) AlarmSummaryDataSeqType
- f) AlarmSummaryInfoSeqType
- g) AlarmSummaryInfoType
- h) AlarmSynchronizationDataSeqType
- i) AlarmSynchronizationInfoChoice
- j) AlarmSynchronizationInfoType
- k) BeginTimeType
- l) EndTimeType
- m) IntervalChoice
- n) IntervalType
- o) InvalidObjectInstanceErrorSeqType
- p) ObjectAlarmSummaryType
- q) ObjectListChoice
- r) ObjectListSetType
- s) ObjectListType
- t) ObjectOfReferenceType
- u) PerceivedSeveritySetType
- v) PerceivedSeverityTypeOpt
- w) ProbableCauseSetType
- x) ProbableCauseTypeOpt
- y) RangeOfObjectsType
- z) ScopedCriteriaSeqType
- aa) ScopedCriteriaType

- bb) SimpleObjectListSetType
- cc) StartTimeType
- dd) StopTimeType
- ee) StopTimeType
- ff) SummaryContentsSetType
- gg) SummaryContentsType
- hh) TerminalRDNRRangeType
- ii) TimeIntervalType

D.3.1.4 Notifications

This specification references the following notifications defined in Recommendation X.780 [16]:

- a) attributeValueChange
- b) objectCreation
- c) objectDeletion
- d) stateChange
- e) communicationsAlarm
- f) equipmentAlarm
- g) environmentalAlarm
- h) processingErrorAlarm
- i) qualityOfServiceAlarm

This specification references the following notification defined in Recommendation Q.816 [6]:

- a) channelChange
- b) heartbeat

This specification references the following notification defined in this document:

- a) currentAlarmSummaryReport

D.3.1.5 Methods

This specification references the following methods defined in Recommendation M.3120 [3].

- In Managed Element:
 - a) enableAudibleVisualLocalAlarmGet
 - b) enableAudibleVisualLocalAlarmSet
 - c) resetAudibleAlarm

This specification references the following methods defined in Recommendation Q.816 [6]:

- In Channel Finder:
 - a) list
- In Channel Finder Component:
 - a) register
 - b) unregister
- In Heartbeat:
 - a) periodGet
 - b) periodSet

The get, set, add and remove methods for attributes contained in managed objects referenced in this document are not repeated here.

This specification references the following methods defined in this document.

- In Management Operations Schedule:
 - a) administrativeStateGet
 - b) administrativeStateSet
 - c) affectedObjectClassGet
 - d) affectedObjectClassSet
 - e) affectedObjectInstancesGet
 - f) affectedObjectInstancesSet
 - g) beginTimeGet
 - h) beginTimeSet
 - i) endTimeGet
 - j) endTimeSet
 - k) endTimeSetDefault
 - l) intervalGet
 - m) intervalSet
 - n) operationalStateGet
- In Current Alarm Summary Control:
 - a) alarmStatusListAdd
 - b) alarmStatusListGet
 - c) alarmStatusListRemove
 - d) alarmStatusListSet
 - e) objectListAdd
 - f) objectListGet
 - g) objectListRemove
 - h) objectListSet
 - i) perceivedSeverityListAdd
 - j) perceivedSeverityListGet
 - k) perceivedSeverityListRemove
 - l) perceivedSeverityListSet
 - m) probableCauseListAdd
 - n) probableCauseListGet
 - o) probableCauseListRemove
 - p) probableCauseListSet
 - q) retrieveCurrentAlarmSummary
- In Enhanced Current Alarm Summary Control:
 - a) alarmSynchronization
- In AlarmSynchronizationDataIterator:
 - a) destroy
 - b) getNext

D.3.1.6 Exceptions

This specification references the following exceptions specified in Recommendation M.3120 [3]:

- a) NOAudibleVisualLocalAlarmPackage

This specification references the following exceptions specified in Recommendation X.780 [16]:

- a) ApplicationError
- b) CreateError
- c) DeleteError

This specification references the following exceptions specified in Recommendation Q.816 [6]:

- a) InvalidFilter
- b) InvalidParameter

- c) FilterComplexityLimit

This specification references the following exception specified in this document (see section 10.4):

- a) InvalidObjectInstanceErrorParameter
- b) NOmanagementOperationsScheduleOperationalStatePkg
- c) SelectionCriteriaNotSupported

D.3.1.7 Name Bindings

This specification references the following name bindings specified in this document (see section 10.9):

- a) CurrentAlarmSummaryControl_ManagedElement
- b) EnhancedCurrentAlarmSummaryControl_ManagedElement
- c) EnhancedCurrentAlarmSummaryControl_ManagedElementComplex
- d) EnhancedCurrentAlarmSummaryControl_Network
- e) ManagementOperationsSchedule_ManagedElement

TABLE OF CONTENTS

1. INTRODUCTION	2
2. REFERENCES	2
3. PROPOSAL	3
3.1 Use Of OMG Notification Service Instead Of Event Reporting	3
3.2 Use Of OMG Telecom Log Service Instead Of Log	5
3.3 Different top Managed Object	6
3.4 Remove Naming Attributes	6
3.5 Remove Cancel Alarm Synchronization Action	6
3.6 No Longer Require Parameter Re-Definition	6
3.7 Suspect Object List Moved To Recommendation X.780	6
3.8 Changes To Alarm Info	6
3.9 Action Results Without The Use Of Linked Replies	7
3.10 Use Of Channels Instead Of EFD Associations	7
3.11 Use Of CORBA Naming Conventions	7
3.12 ASN.1 Definitions Not Used In Recommendation Q.821	7
3.13 Changes To Audible Visual Local Alarm Handling	7
3.14 Heartbeat Service	8
3.15 Complex Constants	8
3.16 CMISE Services Different From CORBA Services	8
4. SCOPE AND PURPOSE	9
4.1 Scope	9
4.2 Purpose	9
4.3 Definitions From Other Documents	9
4.4 Abbreviations	9
5. CONVENTIONS	10
6. ALARM SURVEILLANCE MANAGEMENT INFORMATION	10
6.1 Managed Object Classes	10
6.2 Support Object Classes	10
6.2.1 Current Alarm Summary Control	11
6.2.2 Management Operations Schedule	12
7. ALARM SYNCHRONIZATION	12
7.1 Overview Of Alarm Synchronization	12
7.1.1 Introduction	13
7.1.2 Current Alarms	13
7.1.3 Itemized Alarm Synchronization Requirements	13
7.1.4 Other Information	14
7.2 Alarm Synchronization Information Model	14
7.2.1 Alarm Synchronization Model Overview	14
7.2.2 Alarm Synchronization Managed Object Class	15
7.2.2.1 EnhancedCurrentAlarmSummaryControl	15
7.2.3 Alarm Synchronization Inheritance Hierarchy	15
7.2.4 Name Binding Strategies	16
7.2.4.1 Naming Tree Hierarchy	16
7.2.4.2 Object Creation and Deletion Strategy	16

7.2.4.3	EnhancedCurrentAlarmSummaryControl_ManagedElement	17
7.2.4.4	EnhancedCurrentAlarmSummaryControl_ManagedElementComplex.....	17
7.2.4.5	EnhancedCurrentAlarmSummaryControl_Network	17
7.2.5	Method.....	17
7.2.5.1	alarmSynchronization	17
7.2.5.1.1	Alarm Synchronization Selection Criteria	17
7.2.5.1.2	Structured Event, Event Batch And Typed Event Mapping.....	18
7.2.5.1.3	Alarm Synchronization Iterator.....	20
7.2.5.1.4	Alarm Synchronization Exceptions.....	22
7.2.5.1.5	Alarm Synchronization Parameters	22
7.2.5.1.6	Get Next Parameters	24
7.2.5.1.7	Destroy Parameters	26
7.2.6	Notifications.....	26
7.2.6.1	Object Creation/Object Deletion	26
7.3	Alarm Synchronization Functional Units	26
8.	ALARM SYNCHRONIZATION RELATIONSHIP WITH OTHER DOCUMENTS	26
8.1	Relationship With Recommendation M.3120.....	26
8.2	Relationship With Recommendation X.733	26
8.3	Relationship With OMG Telecom Log Service.....	26
8.4	Relationship With OMG Notification Service	27
9.	CONFORMANCE.....	27
9.1	General Conformance Class Requirement.....	27
9.1.1	Static Conformance	27
9.1.2	Dynamic Conformance.....	27
9.2	Dependent Conformance Class Requirement.....	27
9.2.1	Static Conformance	28
9.2.2	Dynamic Conformance.....	28
9.2.3	Conformance To Support Managed Object Definition.....	28
10.	RECOMMENDATION Q.821 IDL LISTING.....	28
10.1	Imports	29
10.2	Forward Declarations	30
10.3	Structures And Typedefs	30
10.4	Exceptions.....	38
10.5	Current Alarm Summary Control.....	38
10.6	Management Operations Schedule.....	41
10.7	Enhanced Current Alarm Summary Control.....	44
10.8	Notifications	47
10.9	Name Binding.....	47
APPENDIX A - INTEGRATING ALARM SYNCHRONIZATION WITH ALARM CLEARING.....		50
Alarm Clearing Clarified		50
Alarm Clearing Examples.....		51
APPENDIX B - ALARM SYNCHRONIZATION RACE CONDITION.....		53
Overview		53
Example		53
Suggestions On Handling This Race Condition		54
APPENDIX C – ALARM SYNCHRONIZATION SELECTION CRITERIA EXAMPLE.....		55
APPENDIX D – ALARM SURVEILLANCE FUNCTIONS, SERVICES AND FUNCTIONAL UNITS.....		57

D.1	Alarm Surveillance Functions.....	57
D.1.1	Alarm Reporting Functions.....	57
D.1.1.1	Report Alarm.....	58
D.1.1.2	Route Alarm Report	58
D.1.1.3	Request Alarm Report Route.....	58
D.1.1.4	Condition Alarm Reporting.....	58
D.1.1.5	Request Alarm Report Control Condition	58
D.1.1.6	Allow/Inhibit Alarm Reporting.....	58
D.1.1.7	Request Alarm Report History	58
D.1.1.8	Delete Alarm Report History	58
D.1.2	Alarm Summary Functions.....	58
D.1.2.1	Report Current Alarm Summary	58
D.1.2.2	Schedule Current Alarm Summary	59
D.1.2.3	Condition Current Alarm Summary	59
D.1.2.4	Request Current Alarm Summary Schedule	59
D.1.2.5	Request Current Alarm Summary Condition.....	59
D.1.2.6	Allow/Inhibit Current Alarm Summary.....	59
D.1.2.7	Request Current Alarm Summary	59
D.1.3	Alarm Event Criteria Functions	59
D.1.3.1	Condition Alarm Event Criteria	59
D.1.3.2	Request Alarm Event Criteria	59
D.1.4	Alarm Indication Functions	59
D.1.4.1	Inhibit/Allow Audible and Visual Alarm Indications	60
D.1.4.2	Request Inhibit/Allow Audible and Visual Local Alarms Indication	60
D.1.4.3	Reset Audible Alarms	60
D.1.5	Log Control Functions.....	60
D.1.5.1	Allow / Inhibit Logging	60
D.1.5.2	Condition Logging	60
D.1.5.3	Request Log Condition	60
D.2	Alarm Surveillance Service Definition.....	60
D.2.1	Kernel Functional Unit.....	63
D.2.1.1	Alarm Reporting Service	64
D.2.1.2	Initiate Event Channel Service	64
D.2.1.3	Terminate Event Channel Service.....	64
D.2.1.4	Set Event Channel Service	64
D.2.1.5	Get Event Channel Service.....	65
D.2.1.6	Event Channel Reporting Service.....	65
D.2.2	Basic Alarm Report Control Functional Unit.....	66
D.2.2.1	Suspend Alarm Reporting Service.....	67
D.2.2.2	Resume Alarm Reporting Service	67
D.2.3	Enhanced Alarm Report Control Functional Unit.....	67
D.2.3.1	Initiate Alarm Reporting Service	68
D.2.3.2	Terminate Alarm Reporting Service.....	68
D.2.3.3	Obtain Event Channels Service.....	68
D.2.4	Alarm Report Retrieval Functional Unit	69
D.2.4.1	Alarm Report Retrieving Service.....	70
D.2.5	Alarm Report Deletion Functional Unit.....	70
D.2.5.1	Alarm Report Deleting Service	71
D.2.6	Current Alarm Summary Reporting Functional Unit	71
D.2.6.1	Current Alarm Summary Reporting Service.....	72
D.2.7	Basic Management Operations Scheduling Functional Unit.....	72
D.2.7.1	Suspend Management Operations Schedule Service.....	73
D.2.7.2	Resume Management Operations Schedule Service	73
D.2.8	Enhanced Management Operations Scheduling Functional Unit.....	73

D.2.8.1	Initiate Management Operations Schedule Service	74
D.2.8.2	Terminate Management Operations Schedule Service.....	74
D.2.8.3	Set Management Operations Schedule Service.....	74
D.2.8.4	Get Management Operations Schedule Service	74
D.2.9	Current Alarm Summary Reporting Control Functional Unit	74
D.2.9.1	Initiate Current Alarm Summary Control Service.....	75
D.2.9.2	Terminate Current Alarm Summary Control Service	75
D.2.9.3	Set Current Alarm Summary Control Service	75
D.2.9.4	Get Current Alarm Summary Control Service	75
D.2.10	Current Alarm Summary Retrieval Functional Unit	76
D.2.10.1	Retrieve Current Alarm Summary Service.....	76
D.2.11	Alarm Event Criteria Management Functional Unit	77
D.2.11.1	Initiate Alarm Severity Assignment Profile Service	77
D.2.11.2	Terminate Alarm Severity Assignment Profile Service.....	78
D.2.11.3	Set Alarm Severity Assignment Profile Service	78
D.2.11.4	Get Alarm Severity Assignment Profile Service.....	78
D.2.12	Alarm Indication Management Functional Unit	78
D.2.12.1	Set Allow And Inhibit Audible and Visual Local Alarms Service.....	78
D.2.12.2	Get Allow And Inhibit Audible and Visual Local Alarms Service	79
D.2.12.3	Reset Audible Alarms Service	79
D.2.13	Basic Log Control Functional Unit	80
D.2.13.1	Suspend Logging Service	80
D.2.13.2	Resume Logging Service	80
D.2.14	Enhanced Log Control Functional Unit	80
D.2.14.1	Initiate Log Service.....	81
D.2.14.2	Terminate Log Service	81
D.2.14.3	Set Log Service	81
D.2.14.4	Get Log Service	82
D.2.14.5	Log Lookup Service	82
D.2.15	Heartbeat Functional Unit	82
D.2.15.1	Heartbeat Reporting Service	82
D.2.15.2	Set Heartbeat Period Service.....	83
D.2.15.3	Get Heartbeat Period Service	84
D.3	Protocol Specification	84
D.3.1	Elements Of Procedure.....	84
D.3.1.1	Managed Objects	84
D.3.1.2	Valuetypes	85
D.3.1.3	Types	85
D.3.1.4	Notifications.....	87
D.3.1.5	Methods	87
D.3.1.6	Exceptions	88
D.3.1.7	Name Bindings.....	89

LIST OF TABLES

Table 7-1. Alarm Synchronization Method Parameters	24
Table 7-2. Get Next Method Parameters	26
Table A.1. Sample Non-Cleared Alarms	51
Table B.1. Race Condition Example #1.....	53
Table B.2. Race Condition Example #2.....	53
Table C.1. Current Alarm Distribution For Alarm Synchronization Example.....	55
Table D.1. Alarm Reporting Functions and CORBA Services	58
Table D.2. Alarm Summary Functions and CORBA Services	58
Table D.3. Alarm Event Criteria Functions and CORBA Services	59
Table D.4. Alarm Indication Functions and CORBA Services	60
Table D.5. Log Control Functions and CORBA Services	60
Table D.6. Alarm Surveillance Functional Units, Services, Object Classes and Functions.....	63
Table D.7. Event Channel Reporting Service Parameters.....	66
Table D.8. Obtain Event Channels Service.....	69
Table D.9. Current Alarm Summary Reporting Service Parameters	72
Table D.10. Retrieve Current Alarm Summary Service Parameters	76
Table D.11. Set Allow and Inhibit Audible and Visual Local Alarms Service.....	79
Table D.12. Get Allow and Inhibit Audible and Visual Local Alarms Service	79
Table D.13. Reset Audible Alarms Service.....	80
Table D.14. Heartbeat Reporting Service Parameters	83
Table D.15. Set Heartbeat Period Service	83
Table D.16. Get Heartbeat Period Service	84

LIST OF FIGURES

Figure 3-1. OMG Notification Service 4
Figure 3-2. Subset Of OMG Notification Service Objects 4
Figure 3-3. OMG Telecom Log Service 5
Figure 6-1. Containment Relationship Between Alarm Surveillance Support Objects 11
Figure 7-1. Overview of Alarm Synchronization..... 15
Figure 7-2. Alarm Synchronization Inheritance Hierarchy 15
Figure 7-3. Alarm Synchronization Naming Tree Hierarchy 16
Figure 7-4. Mapping Notifications To Structured Events 19
Figure 7-5. Mapping Typed Events To Structured Events 20

Figure C.1. Sample Naming Tree For Alarm Synchronization Example 55

Figure D.1. Kernel Functional Unit 64
Figure D.2. Basic Alarm Report Control Functional Unit 67
Figure D.3. Enhanced Alarm Report Control Functional Unit 68
Figure D.4. Alarm Report Retrieval Functional Unit 70
Figure D.5. Alarm Report Deletion Functional Unit 71
Figure D.6. Current Alarm Summary Reporting Functional Unit 71
Figure D.7. Basic Management Operations Scheduling Functional Unit 73
Figure D.8. Enhanced Management Operations Scheduling Functional Unit 74
Figure D.9. Current Alarm Summary Reporting Control Functional Unit 75
Figure D.10. Current Alarm Summary Retrieval Functional Unit 76
Figure D.11. Alarm Event Criteria Management Functional Unit 77
Figure D.12. Alarm Indication Management Functional Unit 78
Figure D.13. Basic Log Control Functional Unit 80
Figure D.14. Enhanced Log Control Functional Unit 81
Figure D.15. Heartbeat Functional Unit 82