INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.821.1

(09/2001)

SERIES Q: SWITCHING AND SIGNALLING

Q3 interface

CORBA-based TMN alarm surveillance service

ITU-T Recommendation Q.821.1

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4 AND No. 5	Q.120–Q.249
SPECIFICATIONS OF SIGNALLING SYSTEM No. 6	Q.250–Q.309
SPECIFICATIONS OF SIGNALLING SYSTEM R1	Q.310–Q.399
SPECIFICATIONS OF SIGNALLING SYSTEM R2	Q.400–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.799
Q3 INTERFACE	**Q.800–Q.849**
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1699
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000	Q.1700–Q.1799
SPECIFICATIONS OF SIGNALLING RELATED TO BEARER INDEPENDENT CALL CONTROL (BICC)	Q.1900–Q.1999
BROADBAND ISDN	Q.2000–Q.2999

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

ITU-T Recommendation Q.821.1

CORBA-based TMN alarm surveillance service

Summary

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). This Recommendation provides a Stage 2 and Stage 3 Description for Alarm Surveillance to support the associated TMN management service component described in ITU-T Rec. M.3400.

In this Recommendation, both fine-grained and coarse-grained (i.e. facade) interfaces are defined for the alarm management functions. The fine-grained and facade interfaces provide the same support for alarm management. Both can be used with fine-grained and coarse-grained managed objects (e.g. those defined in ITU-T Rec. M.3120).

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.

Source

ITU-T Recommendation Q.821.1 was prepared by ITU-T Study Group 4 (2001-2004) and approved under the WTSA Resolution 1 procedure on 29 September 2001.

FOREWORD

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

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

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

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

NOTE

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

INTELLECTUAL PROPERTY RIGHTS

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

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

CONTENTS

TABLE OF TABLES

TABLE OF FIGURES

Table of CORBA Interfaces

Fine-Grained Interface	Clause	Facade Interface	Clause
CurrentAlarmSummaryControl	10.5.1	CurrentAlarmSummaryControl_F	10.5.2
ManagementOperationsSchedule	10.6.1	ManagementOperationsSchedule_F	10.6.2
EnhancedCurrentAlarmSummaryControl	10.7.1	EnhancedCurrentAlarmSummaryControl_F	10.7.2

ITU-T Recommendation Q.821.1

CORBA-based TMN alarm surveillance service

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]. This Recommendation provides a Stage 2 and Stage 3 Description [5] for Alarm Surveillance to support the associated TMN management service component described in ITU-T Rec. M.3400 [4].

In this Recommendation, both fine-grained and coarse-grained (i.e. facade) interfaces are defined for the alarm management functions. The fine-grained and facade interfaces provide the same support for alarm management. Both can be used with fine-grained and coarse-grained managed objects (e.g. those defined in ITU-T Rec. M.3120).

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.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

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

[2] ITU-T Recommendation M.3100 (1995), *Generic network information model,* plus Amendment 3 (2001): *Definition of the management interface for a generic alarm reporting control (ARC) feature.*

[3] ITU-T Recommendation M.3120 (2001), *CORBA generic network and NE level information model*.

[4] ITU-T Recommendation M.3400 (2000), *TMN Management Functions.*

[5] ITU-T Recommendation Q.68 (1993), *Overview of methodology for developing management services.*

[6] ITU-T Recommendation Q.816 (2001), *CORBA-based TMN services*.

[7] ITU-T Recommendation Q.816.1 (2001), *CORBA-based TMN services: Extensions to support coarse-grained interfaces*.

[8] ITU-T Recommendation Q.821 (2000), *Stage 2 and Stage 3 description for the Q3 interface – Alarm Surveillance*.

[9] ITU-T Recommendation X.701 (1997), *Information technology – Open Systems Interconnection – Systems management overview.*

[10] ITU-T Recommendation X.710 (1997), *Information technology – Open Systems Interconnection – Common management information service*.

[11] ITU-T Recommendation X.722 (1992), *Information technology – Open Systems Interconnection – Structure of Management Information: Guidelines for the definition of managed objects.*

[12] ITU-T Recommendation X.731 (1992), *Information technology – Open Systems Interconnection – Systems Management: State management function.*

[13] ITU-T Recommendation X.733 (1992), *Information technology – Open Systems Interconnection – Systems Management: Alarm reporting function.*

[14] ITU-T Recommendation X.733 (1992), *Information technology – Open Systems Interconnection – Systems Management: Alarm reporting function,* plus Technical Corrigendum 2 (1999).

[15] ITU-T Recommendation X.734 (1992), *Information technology – Open Systems Interconnection – Systems Management: Event report management function.*

[16] ITU-T Recommendation X.734 (1992), *Information technology – Open Systems Interconnection – Systems Management: Event report management function,* plus Amendment 3 (Draft).

[17] ITU-T Recommendation X.735 (1992), *Information technology – Open Systems Interconnection – Systems Management: Log control function*.

[18] ITU-T Recommendation X.780 (2001), *TMN guidelines for defining CORBA managed objects*.

[19] ITU-T Recommendation X.780.1 (2001), *TMS guidelines for defining coarse-grained CORBA managed object interfaces*.

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

[21] OMG Document formal/2001/03/01, *Event Service version 1.1*.

[22] OMG Document formal/2000-06-20, *Notification service, version 1.0*.

[23] OMG Document formal/2000-01-04, *Telecoms Log Service, version 1.0*.

3 Definitions

Definitions from other Recommendations:

Agent	[1]
Alarm	[13]
Alarm Event	[8]

Alarm Info	[18]
Alarm Reporting	[13]
Alarm Status	[8]
Alarm Surveillance	[8]
Alarm Synchronization	[8]
Correlated Notifications	[13]
Current Alarm	[8]
Element Management Layer	[1]
Fault Management	[4]
Inheritance Hierarchy	[18]
Managed Object Class	[9]
Management Information Model	[1]
Manager	[1]
Naming Tree	[18]
Network Management Layer	[1]
Notification Identifier	[13]
Subordinate Objects	[18]
Superior Object	[18]

4 Abbreviations

This Recommendation uses the following abbreviations:

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
IOR	Interoperable Object Reference
MAPDU	Management Application Protocol Data Unit
MD	Mediation Device
MIB	Management Information Base
MOO	Multiple Object Operation

NE	Network Element
OMG	Object Management Group
ORB	Object Request Broker
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 ITU-T Rec. X.780 [18].

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 clause describes the semantics of management information related to Alarm Surveillance. Changes from ITU-T Rec. Q.821 [8] are documented in Appendix III. Also note that in this Recommendation the term "OMG Notification Service" is used to specifically represent reference [22] and "Notification Service" is used to represent a Q.816 [6]-compliant Notification Service. Q.816-compliant Notification Services all share the use of OMG Notification Service Structured Events and Extended TCL filter grammars.

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].

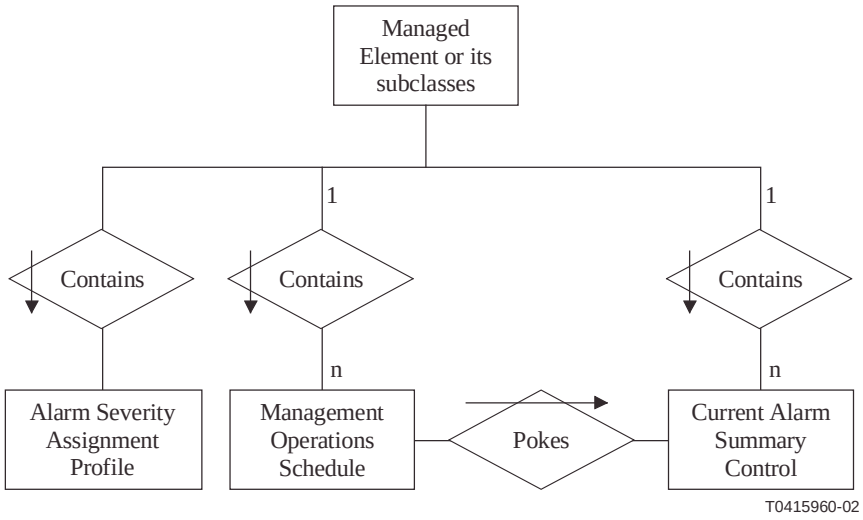

Figure 6-1/Q.821.1 – Containment Relationship between Alarm Surveillance Support Objects

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 ITU-T Rec. X.731 [12]. 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 ITU-T Rec. X.731.

7 Alarm Synchronization

7.1 Overview Of Alarm Synchronization

This clause 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 with 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 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 a Notification Service filter.

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 I/Q.821 [8].

In this Recommendation, 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 clause 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 Recommendation.

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 [12] (as an example, to determine which managed object instances are currently disabled). While this is a problem outside the scope of this Recommendation, 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 ITU-T Rec. X.734 Amendment 3 [16]. 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 ITU-T Rec. X.792 [20]. 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 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.

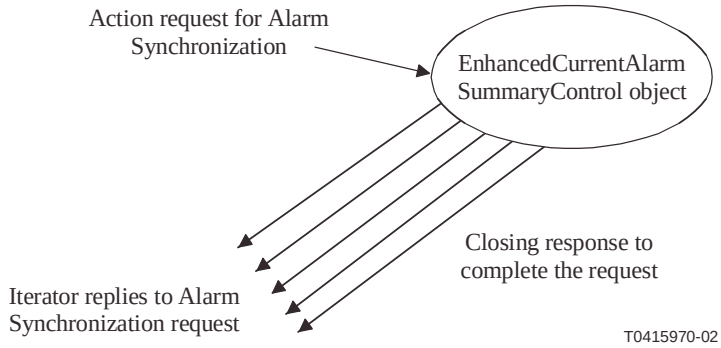

Figure 7-1/Q.821.1 – Overview of Alarm Synchronization

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 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 clause 7. The definitions for the EnhancedCurrentAlarmSummaryControl and EnhancedCurrentAlarmSummaryControl_F interfaces are shown in 10.7.

7.2.3 Alarm Synchronization Inheritance Hierarchy

Figure 7-2 contains the inheritance hierarchy.

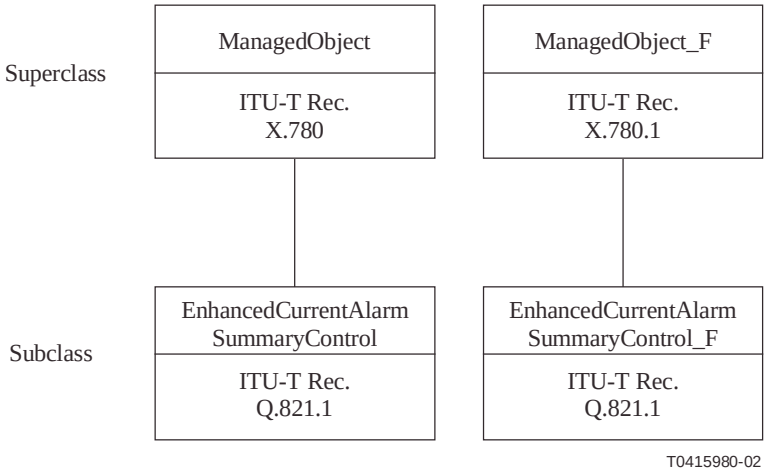

Figure 7-2/Q.821.1 – Alarm Synchronization Inheritance Hierarchy

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 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 ITU-T Rec. M.3120 [3] for 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 required.

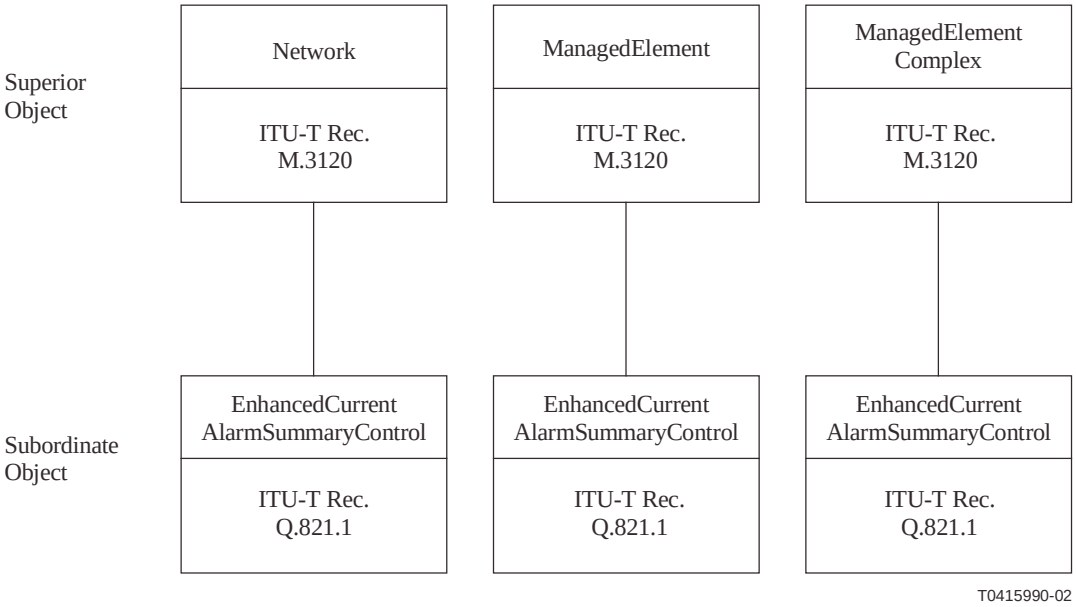

Figure 7-3/Q.821.1 – Alarm Synchronization Naming Tree Hierarchy

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 10.9.

7.2.4.4 EnhancedCurrentAlarmSummaryControl_ManagedElementComplex

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

7.2.4.5 EnhancedCurrentAlarmSummaryControl_Network

The name binding of Enhanced Current Alarm Summary Control to Network is defined in 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 7.2.5.1.3). The definition for the alarmSynchronization action is shown in 10.7.

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 ITU-T Rec. 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 ITU-T Rec. Q.816, the integer-value Scope may be set to:

   – Base managed object only: Scope value of baseObjectOnly. Only the base managed object is included.
   – Whole subtree: Scope value of wholeSubtree. The base managed object and all managed objects contained by the base managed object are included.
   – 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.
   – 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:

   – Scope = baseObjectOnly: Base managed object only.
   – Scope = wholeSubtree: Base managed object and all managed objects directly subordinate to the base manage object.
   – Scope = individualLevel and level = 0: Base managed object only.
   – Scope = individualLevel and level = 1: All managed objects directly subordinate to the base managed object.
   – Scope = baseToLevel and level = 0: Base managed object only.
   – 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 furthered 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 [22] 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 ITU-T Rec. Q.816 is supported (i.e. "MOO 1.0"). Since the Q.816 grammar is a superset of the OMG Notification Service [22] 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:

– Managed Object Class;

– Managed Object Instance;

– Event Type;

– 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 I shows different examples of setting the selection criteria.

7.2.5.1.2 Structured Event, Event Batch And Typed Event Mapping

ITU-T Rec. Q.816 [6] allows an alarm to be distributed either via a Structured Event, an Event Batch or a Typed Event. In this Recommendation, all current alarms are mapped to Structured Events before they are sent via Alarm Synchronization. This clause describes how to perform this mapping for an alarm distributed via Structured Event, Event Batch and Typed Event.

From ITU-T Rec. Q.816, each alarm can be mapped to a Structured Event structure as shown in Figure 7-4.

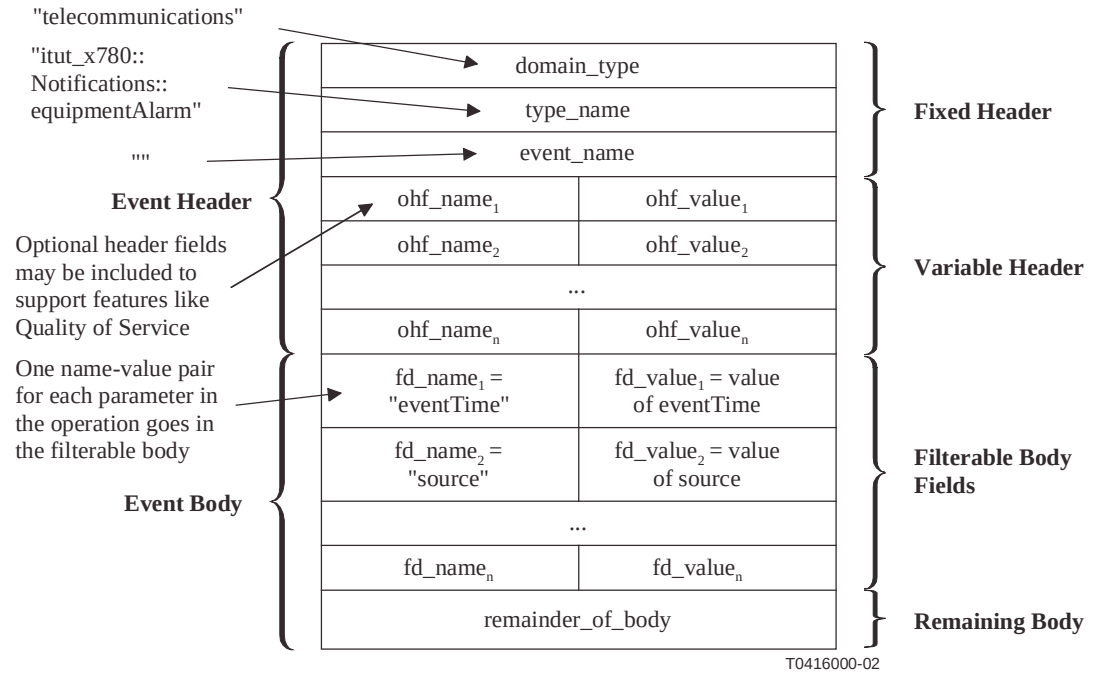

Figure 7-4/Q.821.1 – Mapping Notifications to Structured Events

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 ITU-T Rec. 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.

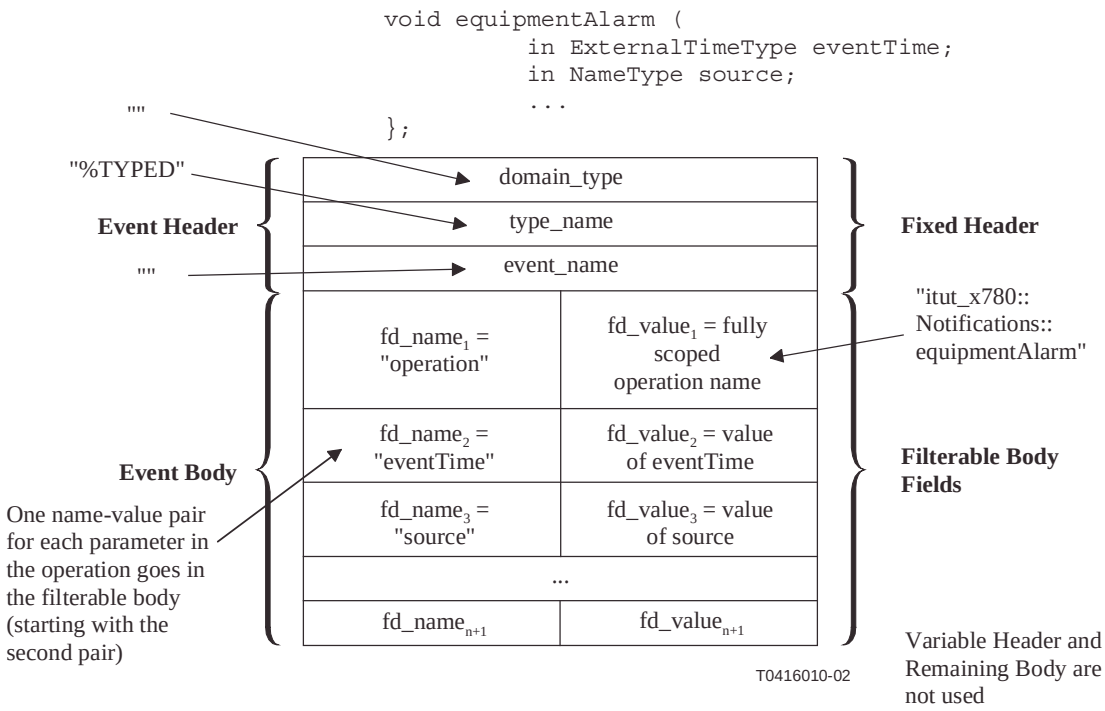

Figure 7-5/Q.821.1 – Mapping Typed Events to Structured Events

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:
   – "itut_x780::Notifications::communicationsAlarm"
   – "itut_x780::Notifications::environmentalAlarm"
   – "itut_x780::Notifications::equipmentAlarm"
   – "itut_x780::Notifications::processingErrorAlarm"
   – "itut_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 n^{th} name element contains the name of the n^{th}-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 n^{th} value element contains the value of the n^{th}–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 ITU-T Rec. X.780 [18]. 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 ITU-T Rec. X.780 and ITU-T Rec. Q.816 [6], the Event Type will be one of the following:

1) "itut_x780::Notifications::communicationsAlarm"
2) "itut_x780::Notifications::environmentalAlarm"
3) "itut_x780::Notifications::equipmentAlarm"
4) "itut_x780::Notifications::processingErrorAlarm"
5) "itut_x780::Notifications::qualityOfServiceAlarm"

This Recommendation 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 Recommendation 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 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 cancelling 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 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 10.4. This will end this request.

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

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

An exception of Application Error type (see ITU-T Rec. X.780 [18]) indicates that some type of application error has occurred. This will end this request.

An exception of Filter Complexity Limit type (see ITU-T Rec. X.780 [18]) indicates that the supplied filter is too complex to process. This will end this request.

7.2.6 Notifications

The following subclauses describe the Alarm Synchronization event notifications. For additional information, see clause 10.

7.2.6.1 Object Creation/Object Deletion

The Object Creation and Object Deletion event notifications, defined in ITU-T Rec. X.780 [18], 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 7.2.4.2.

8 Alarm Synchronization Relationship With Other Documents

This Recommendation uses the service defined in ITU-T Recs. X.780 [18] 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 Recommendation is provided by mechanisms specified in the Notification Service [6] and OMG Telecoms Log Service [23].

8.1 Relationship With ITU-T Rec. M.3120

ITU-T Rec. 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 ITU-T Rec. X.733

ITU-T Rec. X.733 [13] 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 ITU-T Rec. X.780 [18].

Alarm Synchronization totally depends on the Alarm Reporting service. It uses the alarm parameter definitions provided in ITU-T Rec. X.780. In particular, Alarm Synchronization uses the Alarm Reporting definition for alarm clearing (also see Appendix I/Q.821 [8]).

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 II/Q.821.

8.3 Relationship With OMG Telecoms Log Service

OMG Telecoms Log Service [23] 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 [22] defines the Structured Event and Event Batch structure as used for notifications. ITU-T Rec. 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 Compliance And Conformance

This clause defines the criteria that must be met by other standards claiming compliance to this Recommendation and the functions that must be implemented by systems claiming conformance to this Recommendation.

9.1 System Conformance

9.1.1 Conformance Points

This clause describes the conformance points that must be supported by systems claiming conformance to these specifications:

1) An implementation claiming conformance to these requirements must:
   - Support either:
     – The Basic Conformance Profile in ITU-T Rec. Q.816 [6]. In this case, each managed object instance will be an instantiated CORBA object.
     – The Basic Conformance Profile in ITU-T Rec. Q.816.1 [7]. In this case, each supported managed object class will have an instantiated Facade CORBA object (see ITU-T Recs. Q.816.1 and X.780.1 [19]).

- Support either:
  - The Enhanced Current Alarm Summary Control requirements (without Current Alarm Summary Control or Management Operations Schedule support).
  - Both the Enhanced Current Alarm Summary Control and Current Alarm Summary Control requirements (without Management Operations Schedule support).
  - The Enhanced Current Alarm Summary Control, Current Alarm Summary Control and Management Operations Schedule requirements.
  - The Current Alarm Summary Control requirements (without Enhanced Current Alarm Summary Control or Management Operations Schedule support).
  - Both the Current Alarm Summary Control and Management Operations Schedule requirements (without Enhanced Current Alarm Summary Control support).
- Use the IDL listed in clause 10.

2) An implementation claiming conformance to the Enhanced Current Alarm Summary Control requirements must:
   - Support the Enhanced Current Alarm Summary Control managed object specified in 7.2.
   - Support the creation of at least one managed object of the Enhanced Current Alarm Summary Control managed object class.

3) An implementation claiming conformance to the Current Alarm Summary Control requirements must:
   - Support the Current Alarm Summary Control managed object specified in 6.2.1.
   - Support the creation of at least one managed object of the Current Alarm Summary Control managed object class.

4) An implementation claiming conformance to the Management Operations Schedule requirements must:
   - Support the Current Alarm Summary requirements.
   - Support the Management Operations Schedule managed object specified in 6.2.2.
   - Support the creation of at least one managed object of the Management Operations Schedule managed object class.
   - Support the issuing of Current Alarm Summary Control notifications based on Management Operations Schedule.

9.2 Conformance Statement Guidelines

The users of this framework must be careful when writing conformance statements. Because IDL modules are being used as name spaces, they may, as allowed by OMG IDL rules, be split across files. Thus, when a module is extended, its name will not change. Instead, a new IDL file will simply be added. Simply stating the name of a module in a conformance statement, therefore, will not suffice to identify a set of IDL interfaces. The conformance statement must identify a document and year of publication to make sure the right version of IDL is identified.

10 ITU-T Rec. Q.821.1 IDL Listing

```
#ifndef _itut_q821_1_idl_
#define _itut_q821_1_idl_

#include <itut_x780.idl>
#include <itut_x780_1.idl>
#include <itut_x780ct.idl>
#include <itut_m3120.idl>
#include <itut_q816.idl>
```

```
#include <CosNotification.idl>

#pragma prefix "itu.int"

/**
This IDL code (beginning with the line "#ifndef … " through the end of this clause) is
intended to be stored in a file named "itut_q821_1.idl" located in the search path used by
the IDL compiler on your system. A compiler supporting the CORBA version specified in
ITU-T Rec. Q.816 must be used.
*/

/**
This module, itut_q81d1, contains the IDL interface definition for ITU-T Rec. Q.821
(February 2000). The IDL definitions in this file are the object interfaces.
*/

module itut_q821d1
{

/**
```

10.1 Imports

```
*/
     /**
     Types imported from ITU-T Rec. X.780
     */

     typedef itut_x780::AdministrativeStateType AdministrativeStateType;
     typedef itut_x780::DeletePolicyType DeletePolicyType;
     typedef itut_x780::ExternalTimeType ExternalTimeType;
     typedef itut_x780::GeneralizedTimeType GeneralizedTimeType;
     typedef itut_x780::MONameType MONameType;
     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;
     typedef itut_x780ct::TimePeriodType TimePeriodType;

     /**
     Types imported from ITU-T Rec. Q.816
     */

     typedef itut_q816::FilterType FilterType;
     typedef itut_q816::LanguageType LanguageType;
     typedef itut_q816::ScopeType ScopeType;

     /**
     Types imported from ITU-T Rec. M.3120
     */

     typedef itut_m3120::AlarmStatusType AlarmStatusType;

/**
```

10.2 Forward Declarations

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

     interface AlarmSynchronizationDataIterator;
     interface CurrentAlarmSummaryControl;
     interface CurrentAlarmSummaryControl_F;
     interface CurrentAlarmSummaryControlFactory;
```

```
      interface EnhancedCurrentAlarmSummaryControl;
      interface EnhancedCurrentAlarmSummaryControl_F;
      interface EnhancedCurrentAlarmSummaryControlFactory;
      interface ManagementOperationsSchedule;
      interface ManagementOperationsSchedule_F;
      interface ManagementOperationsScheduleFactory;

      /**
      valuetype forward declarations
      */

      valuetype CurrentAlarmSummaryControlValueType;
      valuetype EnhancedCurrentAlarmSummaryControlValueType;
      valuetype ManagementOperationsScheduleValueType;

/**
```

10.3 Structures And Typedefs

```
*/
           /**
           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
      {
           MONameType superiorObjectName;
           TerminalRDNRangeType terminalRDNRange;
      };

      enum ObjectListChoice
      {
           singleObjectChoice,
           rangeOfObjectsChoice
      };

      /**
      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 singleObjectChoice:
          MONameType singleObject;
     case rangeOfObjectsChoice:
          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 cannot 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 MONameType 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 TimeIntervalType from ITU-T Rec. X.780
     */

     /**
     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 ITU-T Rec. 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
{
     MONameType 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 Rec. X.780)
          */
```

```
        FilterType criteria;
            /**
            Default language is "MOO 1.0" (i.e. defaultLanguage in
            ITU-T Rec. X.780)
            */
        LanguageType language;
};

typedef sequence <ScopedCriteriaType> ScopedCriteriaSeqType;

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

        /**
        AlarmSynchronizationInfo ::= CHOICE {
            allObjectsRelativeToSuperior[0] NULL,
            scopedCriteria [1] ScopedCriteria,
            simpleObjectList [2] ObjectList
        }
        */

enum AlarmSynchronizationInfoChoice
{
        allObjectsRelativeToSuperiorChoice,
        scopedCriteriaChoice,
        simpleObjectListChoice
};

union AlarmSynchronizationInfoType
        switch (AlarmSynchronizationInfoChoice)
{
        case scopedCriteriaChoice:
            ScopedCriteriaSeqType scopedCriteria;
        case simpleObjectListChoice:
            SimpleObjectListSetType simpleObjectList;
        /**
        case allObjectsRelativeToSuperiorChoice 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 ITU-T
        Rec. Q.816.
        */

typedef CosNotification::EventBatch AlarmSynchronizationDataSeqType;

        /**
        Note that ITU-T Rec. X.780 uses the ApplicationError exception, so it will be
        used instead of the InvalidBaseMangedObjectError 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 <MONameType> InvalidObjectInstanceErrorSeqType;

      /**
      ITU-T Rec. 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 ITU-T Rec. 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
      */

/**
Define default values for TimePeriodType time types;
*/

const unsigned short defaultDay = 0;
const unsigned short defaultHour = 0;
const unsigned short defaultMinute = 0;
const unsigned short defaultSecond = 0;
const unsigned short defaultMs = 0;
const unsigned short defaultUs = 0;
const unsigned short defaultNs = 0;
```

```
      const unsigned short defaultPs = 0;

      /**
      Constant for packages used in this Recommendation.
      */

      const string ManagementOperationsScheduleOperationalStatePkg =
            "itut_q821d1::ManagementOperationsScheduleOperationalStatePkg";

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

            /**
            "-- see 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 6.2.1 --";;
      */

/**
```

10.5.1 CurrentAlarmSummaryControl Interface

```
*/
      /**
      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 : itut_x780::ManagedObject
      {
            /**
            "-- see 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 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 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 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 II.2.10.1 --"

        @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

/**
```

10.5.2 CurrentAlarmSummaryControl_F Interface

```
*/
    /**
    Current Alarm Summary Control Facade managed object – see ITU-T Rec. X.780.1

    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_F : itut_x780::ManagedObject_F
{
     /**
     "-- see 6.2.1 a) --"

     alarmStatusList GET-REPLACE, ADD-REMOVE
     */

     AlarmStatusSetType alarmStatusListGet
          (in MONameType name)
          raises (itut_x780::ApplicationError);

     void alarmStatusListSet
          (in MONameType name,
          in AlarmStatusSetType alarmStatusList)
          raises (itut_x780::ApplicationError);

     void alarmStatusListAdd
          (in MONameType name,
          in AlarmStatusSetType alarmStatusList)
          raises (itut_x780::ApplicationError);

     void alarmStatusListRemove
          (in MONameType name,
          in AlarmStatusSetType alarmStatusList)
          raises (itut_x780::ApplicationError);

     /**
     "-- see 6.2.1 b) --";;

     objectList GET-REPLACE, ADD-REMOVE
     */

     ObjectListSetType objectListGet
          (in MONameType name)
          raises (itut_x780::ApplicationError);

     void objectListSet
          (in MONameType name,
          in ObjectListSetType objectList)
          raises (itut_x780::ApplicationError);

     void objectListAdd
          (in MONameType name,
          in ObjectListSetType objectList)
          raises (itut_x780::ApplicationError);

     void objectListRemove
          (in MONameType name,
          in ObjectListSetType objectList)
          raises (itut_x780::ApplicationError);

     /**
     "-- see 6.2.1 c) --";;

     perceivedSeverityList GET-REPLACE, ADD-REMOVE
     */

     PerceivedSeveritySetType perceivedSeverityListGet
          (in MONameType name)
          raises (itut_x780::ApplicationError);

     void perceivedSeverityListSet
          (in MONameType name,
          in PerceivedSeveritySetType perceivedSeverityList)
          raises (itut_x780::ApplicationError);
```

```
        void perceivedSeverityListAdd
             (in MONameType name,
             in PerceivedSeveritySetType perceivedSeverityList)
             raises (itut_x780::ApplicationError);

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

        /**
        "-- see 6.2.1 d) --"

        probableCauseList GET-REPLACE, ADD-REMOVE
        */

        ProbableCauseSetType probableCauseListGet
             (in MONameType name)
             raises (itut_x780::ApplicationError);

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

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

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

        /**
        "-- see II.2.10.1 --"

        @param name               Current Alarm Summary managed object instance name
        @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 MONameType name,
             in SummaryContentsSetType summaryContents,
             out AlarmSummaryDataSeqType alarmSummaryData)
             raises (itut_x780::ApplicationError);

        MANDATORY_NOTIFICATION (
             itut_x780::Notifications, currentAlarmSummaryReport)

    }; // interface CurrentAlarmSummaryControl_F

/**
```

10.5.3 CurrentAlarmSummaryControlFactory Interface

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

     interface CurrentAlarmSummaryControlFactory : itut_x780::ManagedObjectFactory
     {
          itut_x780::ManagedObject create
               (in NameBindingType nameBinding,
               in MONameType superior,
```

```
                in string reqID, // no auto-naming, cannot be null
                out MONameType name,
                in StringSetType packageNameList,
                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 ManagementOperationsScheduleValueType :
                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 TimePeriodType interval;
                // GET-REPLACE
          public OperationalStateType operationalState;
                // conditional
                // managementOperationsScheduleOperationalStatePkg
                // GET
     }; // valuetype ManagementOperationsScheduleValueType

     /**
     "-- see 6.2.2 --";;
     */

/**
```

10.6.1 ManagementOperationsSchedule Interface

```
*/
     /**
     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 : itut_x780::ManagedObject
     {
          /**
          "-- see 6.2.2 a) --"

          administrativeState GET-REPLACE
          */
```

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

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

/**
"-- see 6.2.2 b) --"

affectedObjectClass GET-REPLACE
*/

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

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

/**
"-- see 6.2.2 c) --"

affectedObjectInstances GET-REPLACE
*/

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

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

/**
"-- see 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 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 6.2.2 f) --"
```

```
            interval GET-REPLACE
            */

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

            void intervalSet
                  (in TimePeriodType interval)
                  raises (itut_x780::ApplicationError);

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

            "-- see 6.2.2 g) --"

            operationalState GET
            */

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

      }; // interface ManagementOperationsSchedule

      /**
      "-- see 6.2.2 --";;
      */

/**
```

10.6.2 ManagementOperationsSchedule_F Interface

```
*/
      /**
      Management Operations Schedule Facade managed object – see ITU-T Rec. X.780.1

      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_F : itut_x780::ManagedObject_F
      {
            /**
            "-- see 6.2.2 a) --"

            administrativeState GET-REPLACE
            */

            AdministrativeStateType administrativeStateGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

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

            /**
            "-- see 6.2.2 b) --"

            affectedObjectClass GET-REPLACE
            */

            AffectedObjectClassType affectedObjectClassGet
                  (in MONameType name)
                  raises (itut_x780::ApplicationError);

            void affectedObjectClassSet
```

```
      (in MONameType name,
      in AffectedObjectClassType affectedObjectClass)
      raises (itut_x780::ApplicationError);

/**
"-- see 6.2.2 c) --"

affectedObjectInstances GET-REPLACE
*/

AffectedObjectInstancesType affectedObjectInstancesGet
      (in MONameType name)
      raises (itut_x780::ApplicationError);

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

/**
"-- see 6.2.2 d) --"
first activation at begin time, if present, or else when schedule is created

beginTime GET-REPLACE
*/

BeginTimeType beginTimeGet
      (in MONameType name)
      raises (itut_x780::ApplicationError);

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

/**
"-- see 6.2.2 e) --"

endTime GET-REPLACE
*/

EndTimeType endTimeGet
      (in MONameType name)
      raises (itut_x780::ApplicationError);

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

/**
The default EndTimeType is continual
*/

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

/**
"-- see 6.2.2 f) --"

interval GET-REPLACE
*/

TimePeriodType intervalGet
      (in MONameType name)
      raises (itut_x780::ApplicationError);

void intervalSet
```

```
                (in MONameType name,
                in TimePeriodType interval)
                raises (itut_x780::ApplicationError);

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

           "-- see 6.2.2 g) --"

           operationalState GET
           */

           OperationalStateType operationalStateGet
                (in MONameType name)
                raises (itut_x780::ApplicationError,
                     NOmanagementOperationsScheduleOperationalStatePkg);

     }; // interface ManagementOperationsSchedule_F

/**
```

10.6.3 ManagementOperationsScheduleFactory Interface

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

     interface ManagementOperationsScheduleFactory : itut_x780::ManagedObjectFactory
     {
          itut_x780::ManagedObject create
                (in NameBindingType nameBinding,
                in MONameType superior,
                in string reqID, // no auto-naming, cannot be null
                out MONameType name,
                in StringSetType packageNameList,
                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 TimePeriodType 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, thus the naming attribute is no longer needed.
     */

     valuetype EnhancedCurrentAlarmSummaryControlValueType :
```

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

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

/**
```

10.7.1 EnhancedCurrentAlarmSummaryControl Interface

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

     interface EnhancedCurrentAlarmSummaryControl : itut_x780::ManagedObject
     {
          /**
          "-- see 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
          cannot cancel an alarm synchronization call that is currently under progress.
          Secondly, the application can destroy an iterator once it has a return.

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

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

     }; // interface EnhancedCurrentAlarmSummaryControl

     /**
     "-- see 7.2.2 on model –"
     */
```

```
/**
```

10.7.2 EnhancedCurrentAlarmSummaryControl_F Interface

```
*/
     /**
     Enhanced Current Alarm Summary Control Facade managed object – see
     ITU-T Rec. X.780.1
     */

     interface EnhancedCurrentAlarmSummaryControl_F : itut_x780::ManagedObject_F
     {
          /**
          "-- see 7.2.5 on model --"
          */

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

          @param name                              Enhanced Current Alarm Summary managed
                                                   object instance name
          @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 MONameType name,
               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
          cannot cancel an alarm synchronization call that is currently under progress.
          Secondly, the application can destroy an iterator once it has a return.

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

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

     }; // interface EnhancedCurrentAlarmSummaryControl_F

/**
```

10.7.3 AlarmSynchronizationDataIterator Interface

```
*/
     /**
     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 the CORBA alarm returns allowed 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 7.2.5.1.3 on model --"
     */

     interface AlarmSynchronizationDataIterator
     {
          /**
          This method is used to return the next howMany 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 provide both 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
                                                  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 7.2.4.2 on model --"
     */

/**
```

10.7.4 EnhancedCurrentAlarmSummaryControlFactory Interface

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

     interface EnhancedCurrentAlarmSummaryControlFactory :
itut_x780::ManagedObjectFactory
     {
          itut_x780::ManagedObject create
               (in NameBindingType nameBinding,
               in MONameType superior,
               in string reqID, // no auto-naming, cannot be null
```

```
                out MONameType name,
                in StringSetType packageNameList)
                raises (itut_x780::ApplicationError,
                     itut_x780::CreateError);

     }; // interface EnhancedCurrentAlarmSummaryControl

/**
```

10.8 Notifications

```
*/
     /**
     "-- see II.2.6.1 --"

          @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 =
          "itut_q821d1::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_q821d1::CurrentAlarmSummaryControl";
               const boolean subordinateSubclassesAllowed = TRUE;
               const boolean managerCreatesAllowed = TRUE;
               const DeletePolicyType deletePolicy = itut_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_q821d1::ManagementOperationsSchedule";
               const boolean subordinateSubclassesAllowed = TRUE;
               const boolean managerCreatesAllowed = TRUE;
               const DeletePolicyType deletePolicy = itut_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_q821d1::EnhancedCurrentAlarmSummaryControl";
               const boolean subordinateSubclassesAllowed = TRUE;
               const boolean managerCreatesAllowed = FALSE;
               const DeletePolicyType deletePolicy = itut_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_q821d1::EnhancedCurrentAlarmSummaryControl";
               const boolean subordinateSubclassesAllowed = TRUE;
               const boolean managerCreatesAllowed = FALSE;
               const DeletePolicyType deletePolicy = itut_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_q821d1::EnhancedCurrentAlarmSummaryControl";
               const boolean subordinateSubclassesAllowed = TRUE;
               const boolean managerCreatesAllowed = FALSE;
               const DeletePolicyType deletePolicy = itut_x780::notDeletable;
               const string kind = "EnhancedCurrentAlarmSummaryControl";

          }; // module EnhancedCurrentAlarmSummaryControl_Network

     }; // module NameBinding

}; // module itut_q821d1

#endif //_itut_q821_1_idl_
```

Appendix I

Alarm Synchronization Selection Criteria Example

This appendix will show some examples of the Alarm Synchronization selection criteria. Using ITU-T Rec. M.3120 [3] as an example, let us assume the following naming subtree:

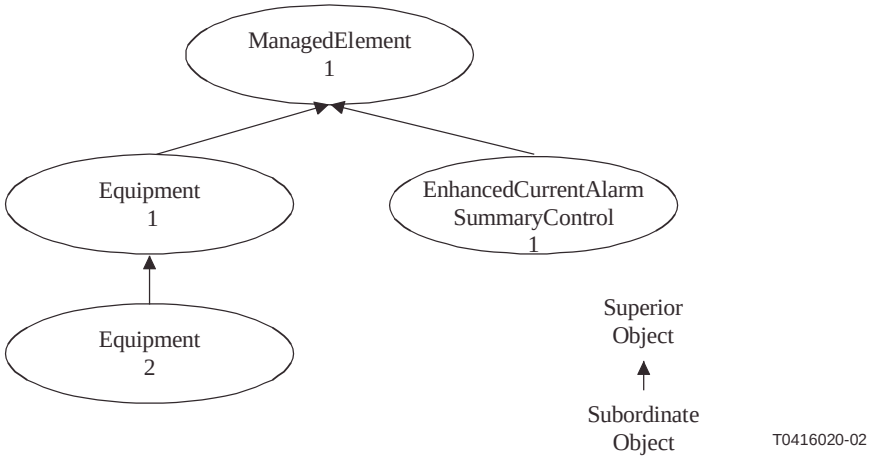

Figure I.1/Q.821.1 – Sample Naming Tree for Alarm Synchronization Example

Let us also assume the following collection of Current Alarms:

Table I.1/Q.821.1 – Current Alarm Distribution for Alarm Synchronization Example

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

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

Individual examples:

1) Alarm Synchronization Info Choice = All Objects Relative To Superior Choice.
   Current Alarms retrieved = A, B and C.
2) Alarm Synchronization Info Choice = Scoped Criteria Choice.
   Base Managed Object = Managed Element 1.
   Scope = Whole subtree.
   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 Choice.
   Base Managed Object = Managed Element 1.
   Scope = Individual Level.

Level = 1.

Criteria = "TRUE".

Current Alarms retrieved = B.

4) Alarm Synchronization Info Choice = Scoped Criteria Choice.

Base Managed Object = Equipment 1.

Scope = Base Object Only.

Criteria = "TRUE".

Current Alarms retrieved = B.

5) Alarm Synchronization Info Choice = Scoped Criteria Choice.

Base Managed Object = Managed Element 1.

Scope = Base To Level.

Level = 1.

Criteria = "TRUE".

Current Alarms retrieved = A, B.

6) Alarm Synchronization Info Choice = Simple Object List Choice.

Object Instance #1 = Equipment 1.

Object Instance #2 = Equipment 2.

Current Alarms retrieved = B and C.

7) Alarm Synchronization Info Choice = Simple Object List Choice.

Object Instance #1 = Equipment = 1.

Object Instance #2 = Equipment = 2.

Object Instance #3 = Equipment = 3 (i.e. outside the scope of this Recommendation).

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

Appendix II

Alarm Surveillance Functions, Services And Functional Units

This appendix updates the clause 1/Q.821 [8], Alarm Surveillance, according to the capabilities available in CORBA. Appendix II outlines the alarm surveillance functions and services available via ITU-T Recs. X.780 [18], Q.816 [6] and this Recommendation.

The general format of Q.821 Alarm Surveillance clause has been maintained. Note, however, that such terms as "functional unit" do not exist in ITU-T Rec. X.780 as they do in Q/CMISE. In ITU-T Rec. X.780, functional units are not negotiated as they are with Q/CMISE. Also, ITU-T Rec. X.780 has added new capabilities, such as Heartbeat Service, and new objects, such as Event Channels. Note that this appendix assumes the use of the OMG Notification Service [22].

II.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 leads to the generation of other management data.

The management information related to Alarm Surveillance (the semantics of which is described) includes managed object classes, support object classes and their associated attributes.

The Q/CMISE services from ITU-T Rec. Q.821 [8] are shown for comparison purposes only and are not discussed further in this Recommendation.

II.1.1 Alarm Reporting Functions

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

Table II.1/Q.821.1 – Alarm Reporting Functions and CORBA Services

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
Request Alarm Report History	Alarm Report Retrieving	Alarm Report Retrieving
Delete Alarm Report History	Alarm Report Deleting	Alarm Report Deleting

II.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.

II.1.1.2 Route Alarm Report

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

II.1.1.3 Request Alarm Report Route

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

II.1.1.4 Condition Alarm Reporting

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

II.1.1.5 Request Alarm Report Control Condition

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

II.1.1.6 Allow/Inhibit Alarm Reporting

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

II.1.1.7 Request Alarm Report History

TMN requests and receives specified historical alarm information.

II.1.1.8 Delete Alarm Report History

TMN requests to delete specified historical alarm information.

II.1.2 Alarm Summary Functions

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

Table II.2/Q.821.1 – Alarm Summary Functions and CORBA Services

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 Alarm 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

Table II.2/Q.821.1 – Alarm Summary Functions and CORBA Services *(concluded)*

Function	CORBA Service	Q/CMISE Service
Request Current Alarm Summary	Retrieve Current Alarm Summary	Retrieve Current Alarm Summary
Request Enhanced Current Alarm Summary	Alarm Synchronization	Alarm Synchronization
Cancel Alarm Synchronization	<Not Applicable>	Cancel Alarm Synchronization

II.1.2.1 Report Current Alarm Summary

Network element provides TMN (based on a predefined schedule) with a Current Alarm Summary.

II.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.

II.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.

II.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.

II.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.

II.1.2.6 Allow/Inhibit Current Alarm Summary

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

II.1.2.7 Request Current Alarm Summary

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

II.1.2.8 Request Enhanced Current Alarm Summary

TMN requests the network element to send an Enhanced Current Alarm Summary alarm synchronization report; network element responds with the summary.

II.1.3 Alarm Event Criteria Functions

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

Table II.3/Q.821.1 – Alarm Event Criteria Functions and CORBA Services

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

II.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.

II.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.

II.1.4 Alarm Indication Functions

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

Table II.4/Q.821.1 – Alarm Indication Functions and CORBA Services

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

II.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.

II.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.

II.1.4.3 Reset Audible Alarms

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

II.1.5 Log Control Functions

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

Table II.5/Q.821.1 – Log Control Functions and CORBA Services

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

II.1.5.1 Allow/Inhibit Logging

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

II.1.5.2 Condition Logging

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

II.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.

II.2 Alarm Surveillance Service Definition

This clause defines the services needed to support the alarm surveillance functions specified in II.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 II.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 [9]. Table II.6 lists these functional units and their corresponding services.

Table II.6/Q.821.1 – Alarm Surveillance Functional Units, Services, Object Classes and 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
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,	Condition Alarm Reporting, Route Alarm Report, Request Alarm Report Route, Request Alarm Report Control Condition

Table II.6/Q.821.1 – Alarm Surveillance Functional Units, Services, Object Classes and Functions *(continued)*

Functional Unit	CORBA Services	Object Classes	Functions
		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	
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
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

Table II.6/Q.821.1 – Alarm Surveillance Functional Units, Services, Object Classes and Functions *(concluded)*

Functional Unit	CORBA Services	Object Classes	Functions
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
Alarm Synchronization	Alarm Synchronization	Enhanced Current Alarm Summary Control	Request Enhanced Current Alarm Summary

II.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 II.1 shows the interactions between the managing and managed system for this functional unit. Note that the Event Channel shown in Figure II.1 may be predefined.

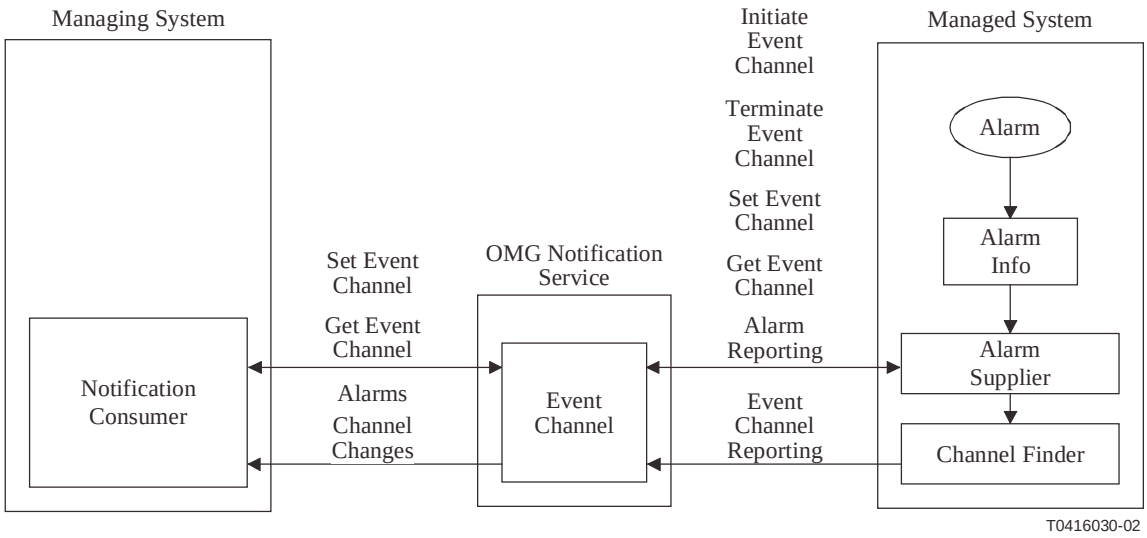

Figure II.1/Q.821.1 – Kernel Functional Unit

II.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 II.1.1.

For the service definition, see OMG Notification Service [22], ITU-T Rec. X.780 [18] and ITU-T Rec. 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.

II.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 II.1.1.

ITU-T Rec. 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 II.2.1.6).

For the service definition, see OMG Notification Service [22].

II.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 II.1.1.

ITU-T Rec. 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 II.2.1.6).

For the service definition, see OMG Notification Service [22].

II.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 II.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 [22].

II.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 II.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 [22].

II.2.1.6 Event Channel Reporting Service

The Event Channel Reporting service allows a managed system to report changes to OMG Notification Service [22] 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 II.1.1.

For the service definition, see ITU-T Rec. Q.816 [6].

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

Table II.7/Q.821.1 – Event Channel Reporting Service Parameters

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Event Header	M	–	
Fixed Event Header	M	–	
Event Type	M	–	
Domain Name	M	–	"telecommunications"
Type Name	M	–	"itut_x780::Notifications::channelChange"
Event Name	M	–	Null
Optional Header Fields	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 And Scopes	M	–	
Base	M	–	
Scope	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	–	

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 And Scopes – The base managed object instances and the scopes of managed object instances below them sending notifications to this channel. An empty list indicates that all base managed objects on the system are covered by this channel.

– Event Types – List of Event Types support by this Event Channel. As an example, it could contain "itut_x780::Notifications::equipmentAlarm" if Equipment Alarms are supported (see 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.

II.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 II.2 shows the interactions between the managing and managed system for this functional unit. Note that the Event Channel object shown in Figure II.2 may be predefined.

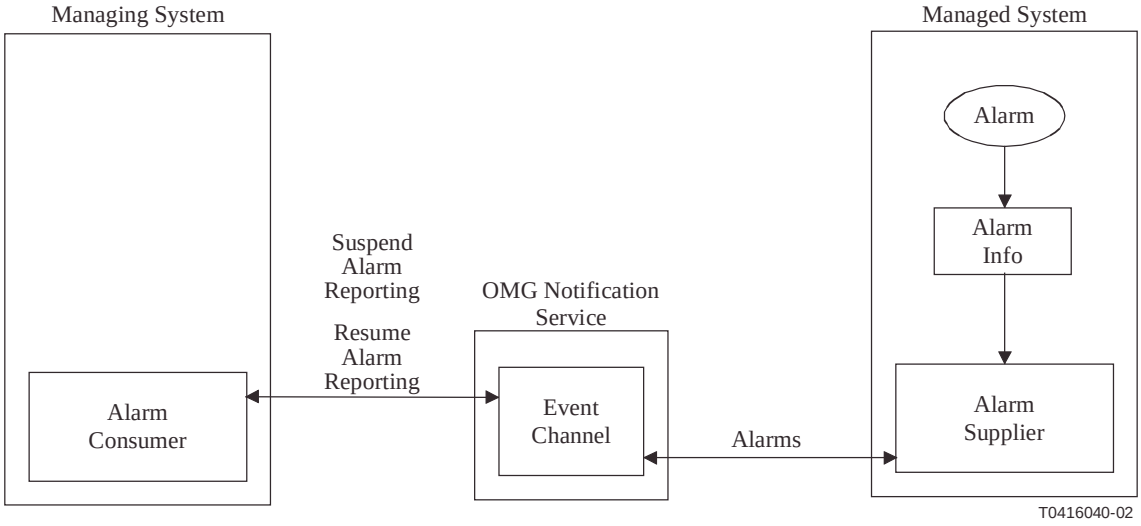

Figure II.2/Q.821.1 – Basic Alarm Report Control Functional Unit

II.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 II.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 [22] under Sequence Proxy Pull Consumer, Sequence Proxy Push Supplier, Structured Proxy Pull Consumer, Structured Proxy Push Supplier, or Typed Proxy Push Supplier.

II.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 II.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 [22] under Sequence Proxy Pull Consumer, Sequence Proxy Push Supplier, Structured Proxy Pull Consumer, Structured Proxy Push Supplier, or Typed Proxy Push Supplier.

II.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 II.3 shows the interactions between the managing and managed system for this functional unit.

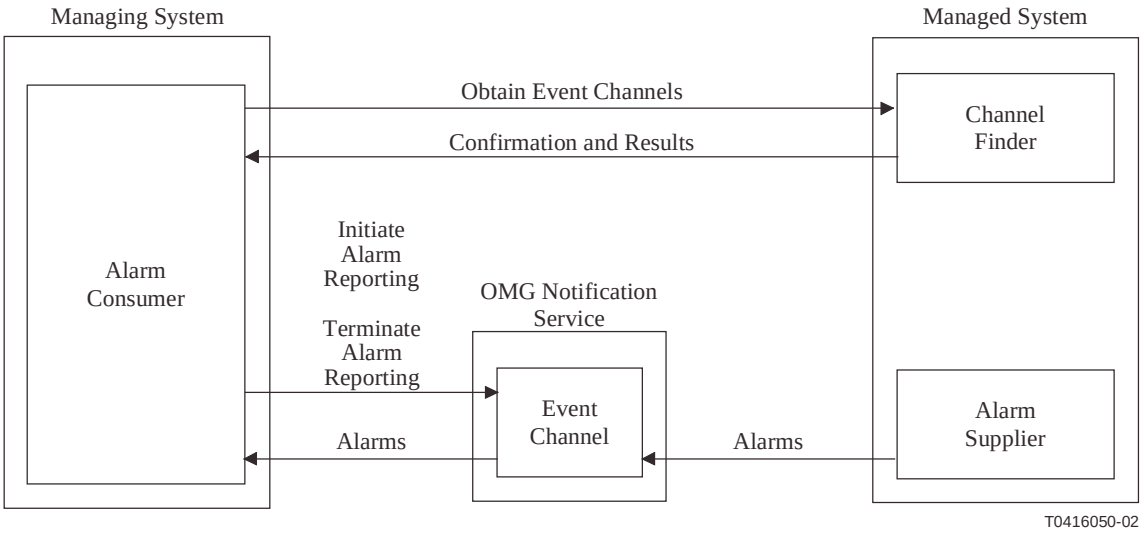

Figure II.3/Q.821.1 – Enhanced Alarm Report Control Functional Unit

II.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 II.1.1.

For the service definition, see OMG Notification Service [22].

II.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 II.1.1.

For the service definition, see OMG Notification Service [22].

II.2.3.3 Obtain Event Channels Service

The Obtain Event Channels service allows a managing system to retrieve what OMG Notification Service [22] 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 II.1.1.

The semantics of the Channel Finder list method are defined in ITU-T Rec. Q.816 [6].

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

Table II.8/Q.821.1 – 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 And Scopes	–	M	
Base	–	M	
Scope	–	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	

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 And Scopes: The base managed object instances and the scopes of managed object instances below them sending notifications to this channel. An empty list indicates that all base managed objects on the system are covered by this channel.

– Event Types: List of Event Types support by this Event Channel. As an example, it could contain "itut_x780::Notifications::equipmentAlarm" if Equipment Alarms are supported (see 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.

Exception

– Application Error.

II.2.4 Alarm Report Retrieval Functional Unit

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

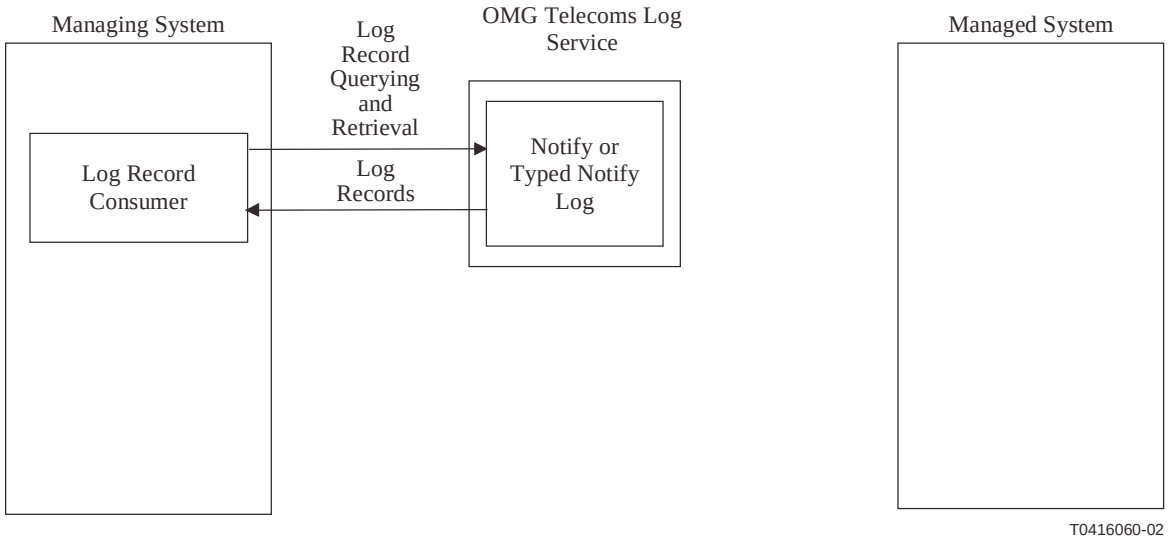

Figure II.4/Q.821.1 – Alarm Report Retrieval Functional Unit

II.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 II.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 Telecoms Log Service [23] Log Records do not have subclassifications (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 Telecoms Log Service Get Record Attribute method. For the service definition in this case, see the OMG Telecoms Log Service.

Alternatively, attributes for multiple alarm Log Records may be retrieved by utilizing the OMG Telecoms 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 Telecoms Log Service.

II.2.5 Alarm Report Deletion Functional Unit

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

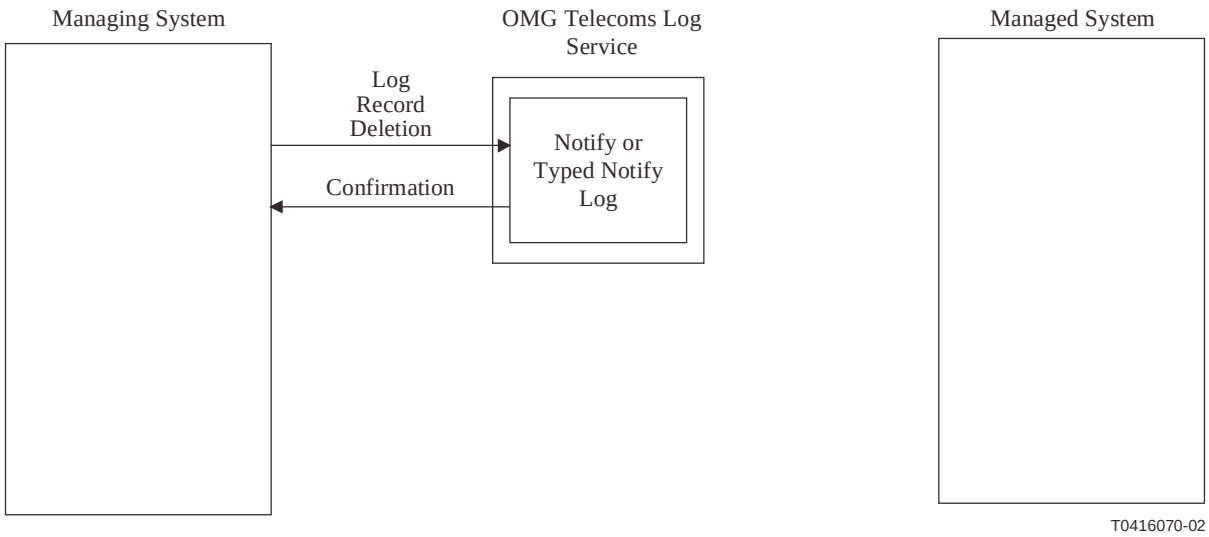

Figure II.5/Q.821.1 – Alarm Report Deletion Functional Unit

II.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 II.1.1.

For the service definition, see OMG Telecoms Log Service [23] under Log Record deletion.

II.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 II.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 II.6 may be predefined. The Management Operations Schedule object shall be present but need not be modifiable by the managing system.

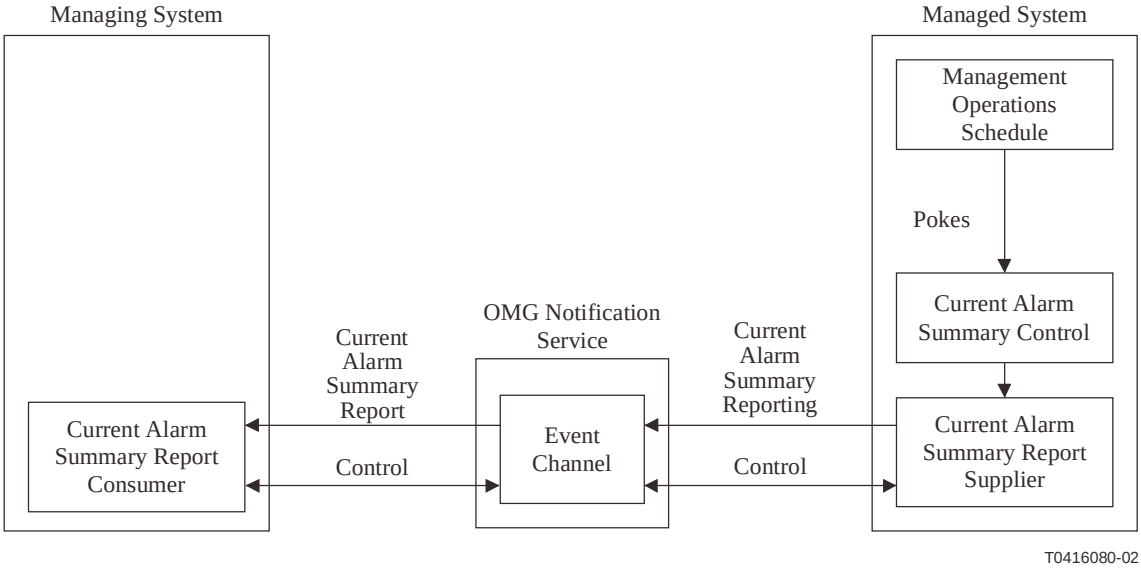

Figure II.6/Q.821.1 – Current Alarm Summary Reporting Functional Unit

II.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 II.1.2.

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

Table II.9/Q.821.1 – Current Alarm Summary Reporting Service Parameters

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Event Header	M	–	
Fixed Event Header	M	–	
Event Type	M	–	
Domain Name	M	–	"telecommunications"
Type Name	M	–	"itut_x780::Notifications::currentAlarm SummaryReport"
Event Name	M	–	Null
Optional Header Fields	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 ObjectAlarm SummaryType
Object Of Reference	C	–	
Summary Info	C	–	
Perceived Severity	C	–	
Alarm Status	C	–	
Probable Cause	C	–	

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);

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

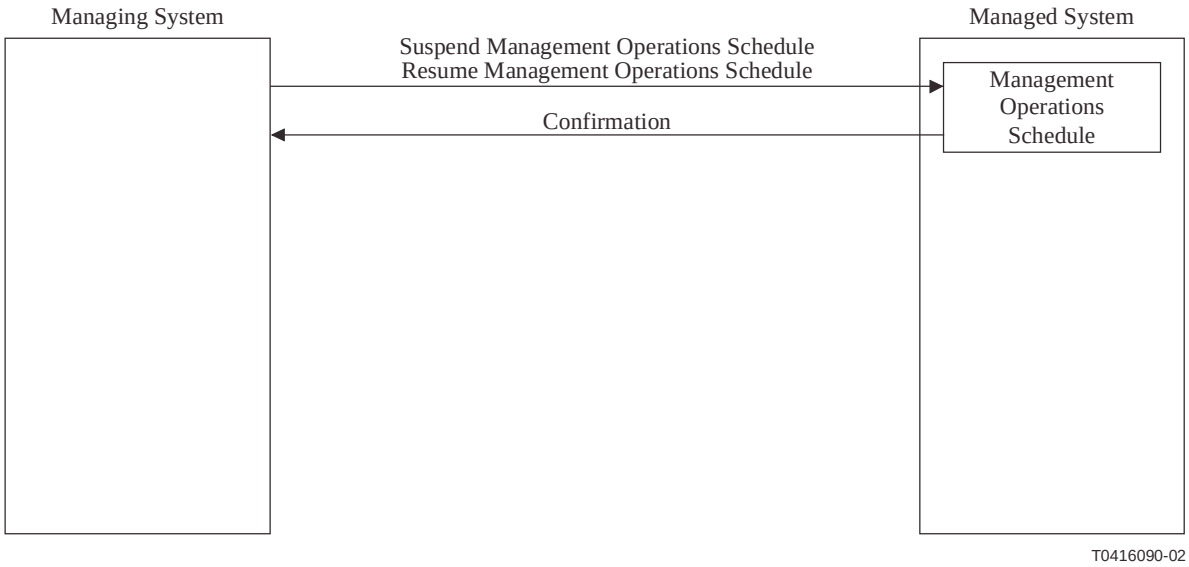

Figure II.7/Q.821.1 – Basic Management Operations Scheduling Functional Unit

II.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 [8]. This service supports the Allow/Inhibit Current Alarm Summary function identified in II.1.2.

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

II.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 [8]. This service supports the Allow/Inhibit Current Alarm Summary function identified in II.1.2.

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

II.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 II.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.

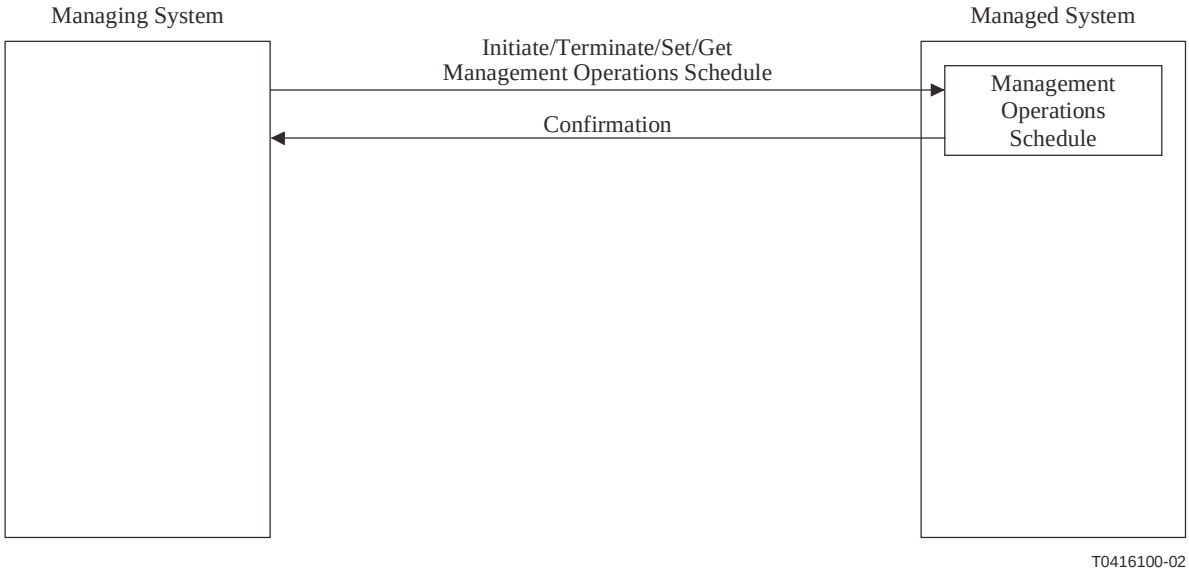

Figure II.8/Q.821.1 – Enhanced Management Operations Scheduling Functional Unit

II.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 [18]. This service supports the Schedule Current Alarm Summary and Route Current Alarm Summary functions identified in II.1.2.

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

II.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 [18]. This service supports the Schedule Current Alarm Summary function identified in II.1.2.

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

II.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 [18]. This service supports the Schedule Current Alarm Summary and Route Current Alarm Summary functions identified in II.1.2.

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

II.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 [18]. This service supports the Request Current Alarm Summary Schedule and Request Current Alarm Summary Route functions identified in II.1.2.

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

II.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 II.9 shows the interactions between the managing and managed system for this functional unit.

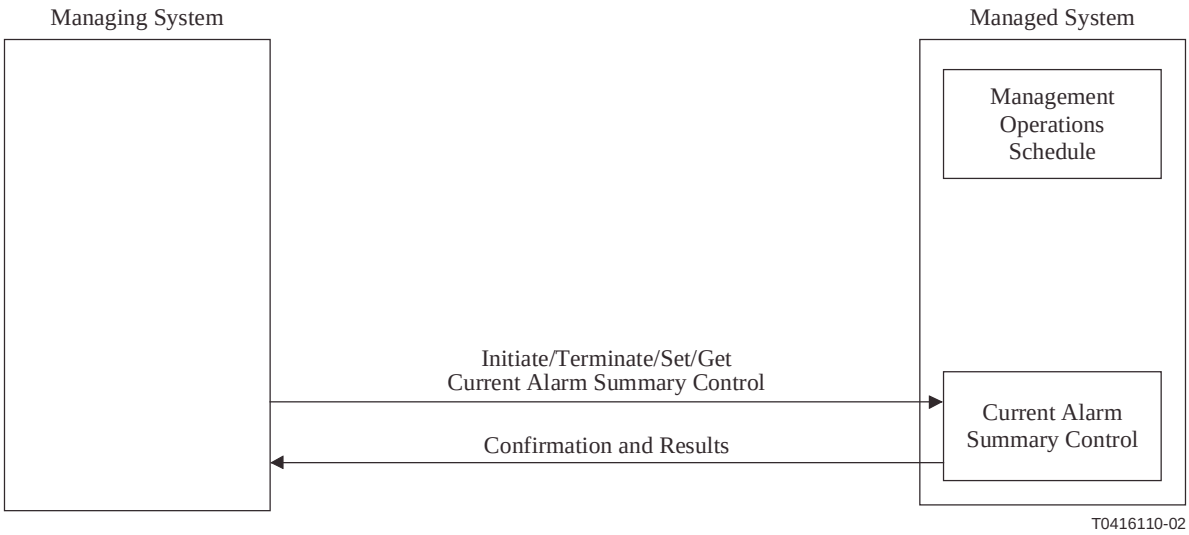

Figure II.9/Q.821.1 – Current Alarm Summary Reporting Control Functional Unit

II.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 [18]. This service supports the Schedule Current Alarm Summary function identified in II.1.2.

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

II.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 [18]. This service supports the Schedule Current Alarm Summary function identified in II.1.2.

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

II.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 [18]. 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 II.1.2.

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

II.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 [18]. This service supports the Request Current Alarm Summary Schedule function identified in II.1.2.

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

II.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 II.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 II.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.

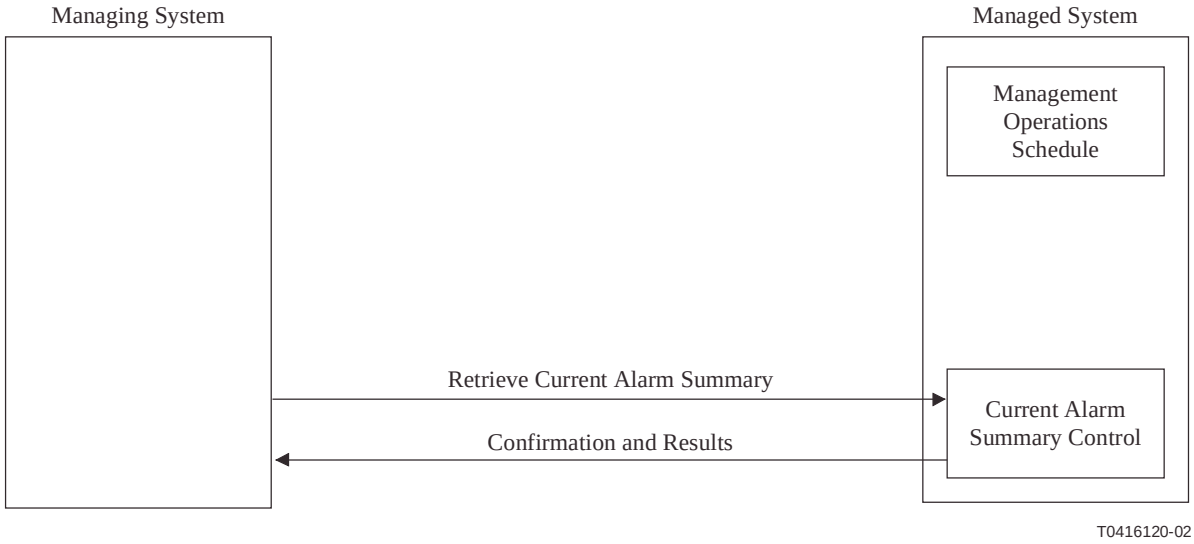

Figure II.10/Q.821.1 – Current Alarm Summary Retrieval Functional Unit

II.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 II.1.2.

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

Table II.10/Q.821.1 – Retrieve Current Alarm Summary Service Parameters

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Name	O	–	Only used if access to the Current Alarm Summary Control managed object is through a Facade object
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	

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).

Exception

– Application Error.

II.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 II.11 shows the interactions between the managing and managed system for this functional unit.

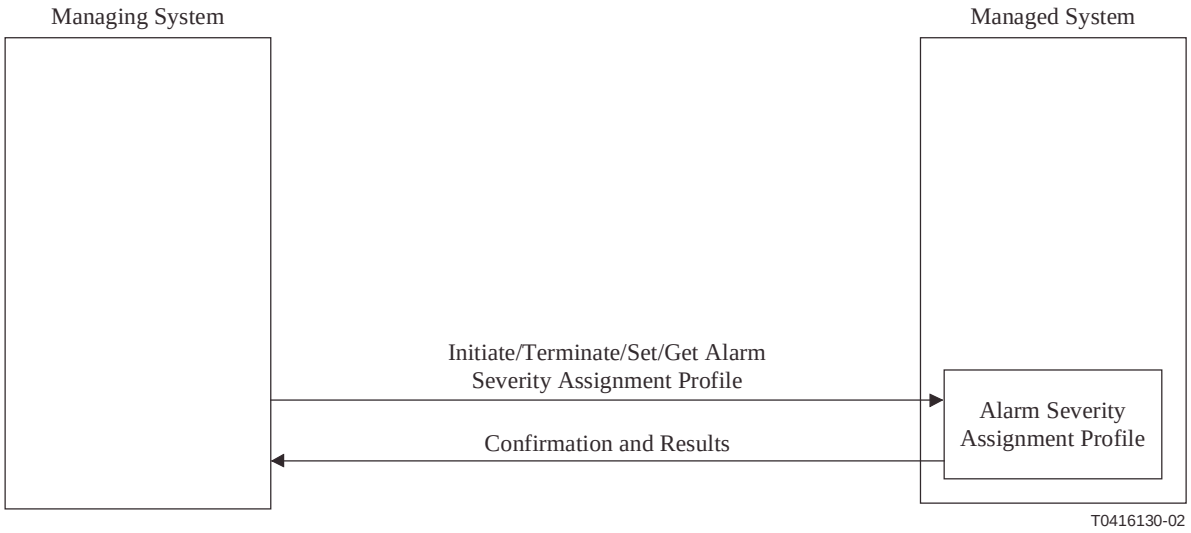

Figure II.11/Q.821.1 – Alarm Event Criteria Management Functional Unit

II.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 [18]. This service supports the Condition Alarm Event Criteria function identified in II.1.3.

The semantics of the Alarm Severity Assignment Profile attributes are defined in [3].

II.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 [18]. This service supports the Condition Alarm Event Criteria function identified in II.1.3.

The semantics of the Alarm Severity Assignment Profile attributes are defined in [3].

II.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 [18]. This service supports the Condition Alarm Event Criteria function identified in II.1.3.

The semantics of the Alarm Severity Assignment Profile object class are described in [3].

II.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 [18]. This service supports the Request Alarm Event Criteria function identified in II.1.3.

The semantics of the Alarm Severity Assignment Profile object class are described in [3].

II.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 II.12 shows the interactions between the managing and managed system for this functional unit.

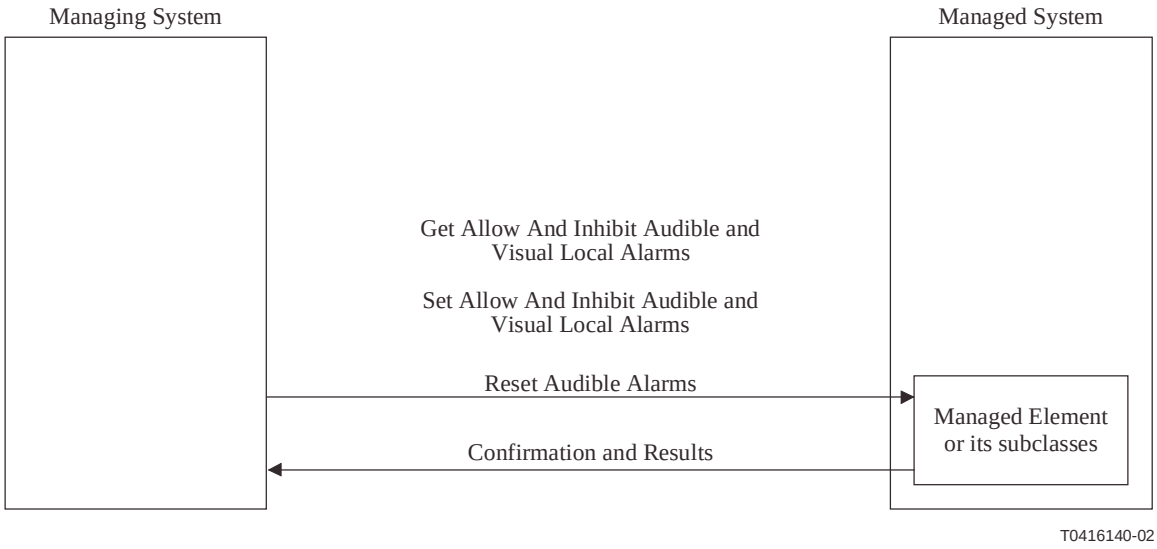

Figure II.12/Q.821.1 – Alarm Indication Management Functional Unit

II.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 II.1.4.

Table II.11 shows the parameters used in the Set Allow and Inhibit Audible and Visual Local Alarms service.

Table II.11/Q.821.1 – Set Allow and Inhibit Audible and Visual Local Alarms Service

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Name	O	–	Only used if access to the Managed Element managed object is through a Facade object
Enable	M	–	
Exceptions	–	P	

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.

II.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 II.1.4.

Table II.12 shows the parameters used in the Get Allow and Inhibit Audible and Visual Local Alarms service.

Table II.12/Q.821.1 – Get Allow and Inhibit Audible and Visual Local Alarms Service

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Name	O	–	Only used if access to the Managed Element managed object is through a Facade object
Results	–	M	
Exceptions	–	P	

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.

II.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 II.1.4.

Table II.13 shows the parameters used in the Reset Audible Alarm service.

Table II.13/Q.821.1 – Reset Audible Alarms Service

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Name	O	–	Only used if access to the Managed Element managed object is through a Facade object
Exceptions	–	P	

The following parameters are defined for use in the Reset Audible Alarms service:

Exceptions

– Application Error;

– No Audible Visual Local Alarm Package.

II.2.13 Basic Log Control Functional Unit

The Basic Log Control functional unit contains the Suspend Logging and the Resume Logging services. Figure II.13 shows the interactions between the managing and managed system for this functional unit. Note that the Log object shown in Figure II.13 may either have a predefined Filter or the Filter may be absent.

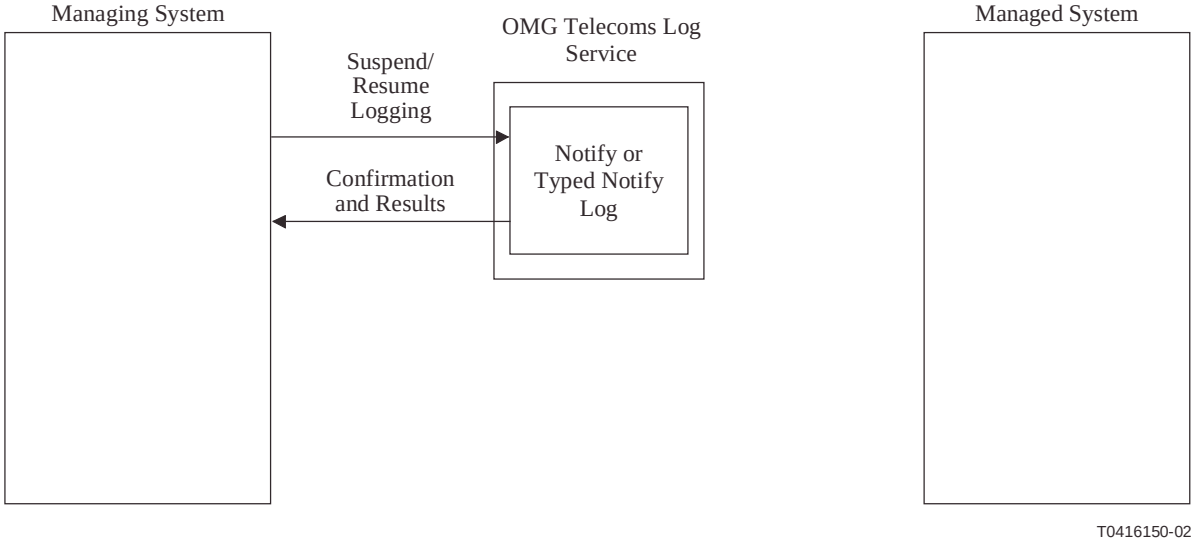

Figure II.13/Q.821.1 – Basic Log Control Functional Unit

II.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 II.1.5.

For the service definition, see OMG Telecoms Log Service [23] and the use of Operational State.

II.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 II.1.5.

For the service definition, see OMG Telecoms Log Service [23] and the use of Operational State.

II.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 II.14 shows the interactions between the managing and managed system for this functional unit. Note that the Log object shown in Figure II.13 may either be predefined or be created using the Initiate Log service.

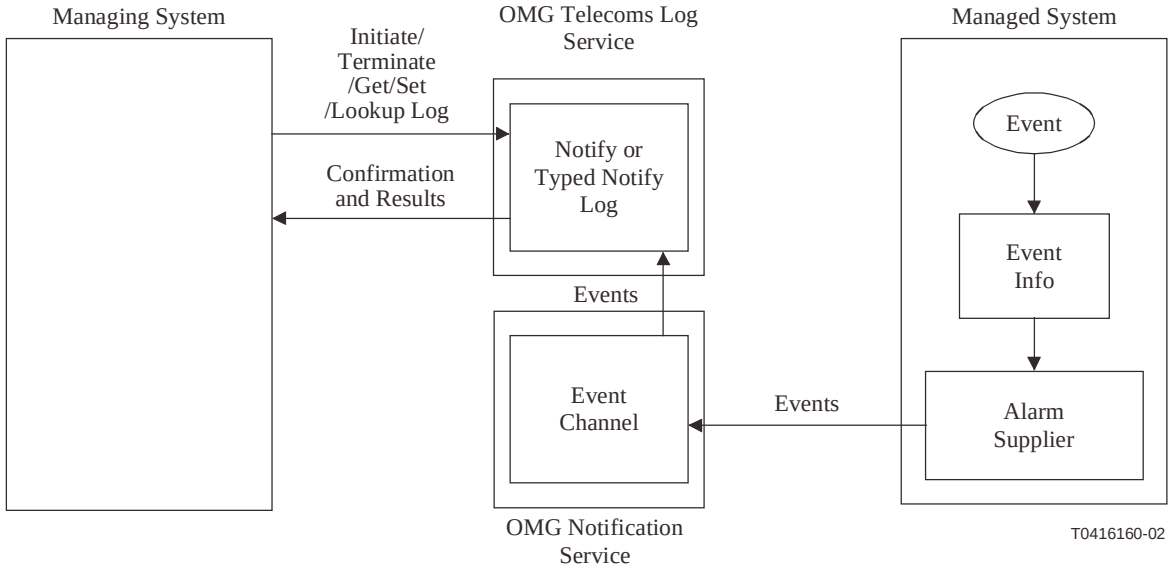

Figure II.14/Q.821.1 – Enhanced Log Control Functional Unit

II.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 Telecoms Log Service [23]. This service supports the Condition Logging function identified in II.1.5.

Log Event Channels must be registered with the Channel Finder service (see ITU-T Rec. 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 Telecoms Log Service notifications may be of a different format from notifications generated from ITU-T Rec. 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 Telecoms Log Service under NotifyLogFactory or TypedNotifyLogFactory.

II.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 Telecoms Log Service [23]. This service supports the Condition Logging function identified in II.1.5.

For the service definition, see OMG Telecoms Log Service under NotifyLogFactory or TypedNotifyLogFactory.

II.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 II.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 Telecoms Log Service [23] under NotifyLog or TypedNotifyLog.

II.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 II.1.5.

For the service definition, see OMG Telecoms Log Service [23] under NotifyLog or TypedNotifyLog.

II.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 II.1.5.

For the service definition, see OMG Telecoms Log Service [23] under NotifyLogFactory.

II.2.15 Heartbeat Functional Unit

The Heartbeat functional unit contains the Heartbeat Reporting, the Get Heartbeat Period and the Set Heartbeat Period services. Figure II.15 shows the interactions between the managing and managed system for this functional unit. Note that the Event Channel shown in Figure II.15 may be predefined.

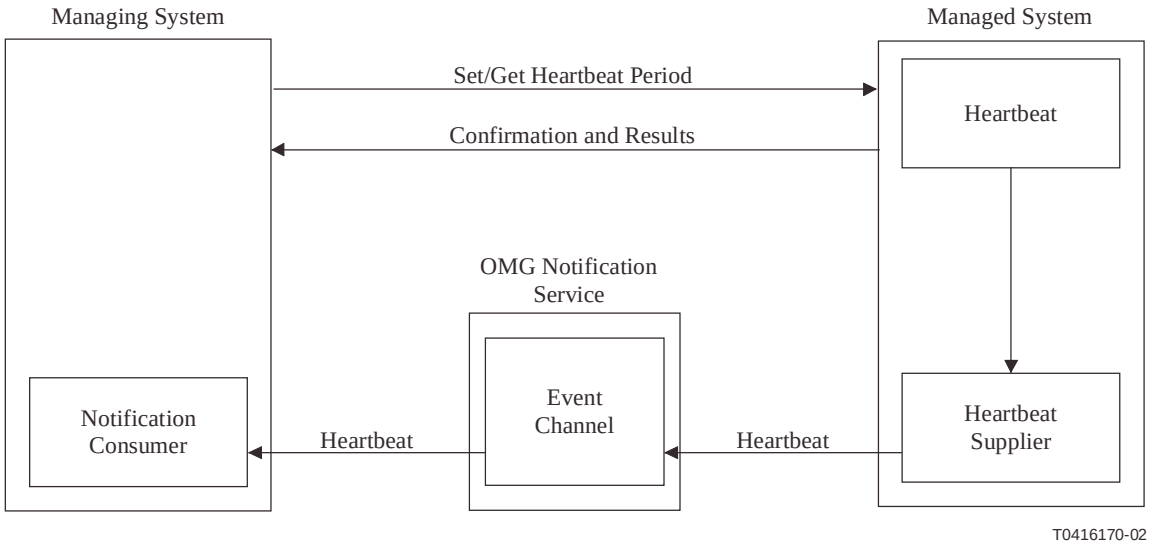

Figure II.15/Q.821.1 – Heartbeat Functional Unit

II.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 II.1.1. For the service definition, see ITU-T Rec. Q.816 [6].

Table II.14 lists the parameters for the Heartbeat Reporting service.

Table II.14/Q.821.1 – Heartbeat Reporting Service Parameters

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Event Header	M	–	
Fixed Event Header	M	–	
Event Type	M	–	
Domain Name	M	–	"telecommunications"
Type Name	M	–	"itut_q816::Notifications::heartbeat"
Event Name	M	–	Null
Optional Header Fields	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		

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.

II.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 II.1.1.

Table II.15 shows the parameters used in the Set Heartbeat Period service.

Table II.15/Q.821.1 – Set Heartbeat Period Service

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Period	M	–	
Exceptions	–	P	

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.

Exception

– Application Error.

II.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 II.1.1.

Table II.16 shows the parameters used in the Get Heartbeat Period service.

Table II.16/Q.821.1 – Get Heartbeat Period Service

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Period	–	M	
Exceptions	–	P	

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.

Exception

– Application Error.

II.2.16 Alarm Synchronization Functional Unit

The Alarm Synchronization functional unit contains the Alarm Synchronization service described below. Figure II.16 shows the interactions between the managing and managed system for this functional unit. Note that the Enhanced Current Alarm Summary Control object(s) shown in Figure II.16 may be predefined.

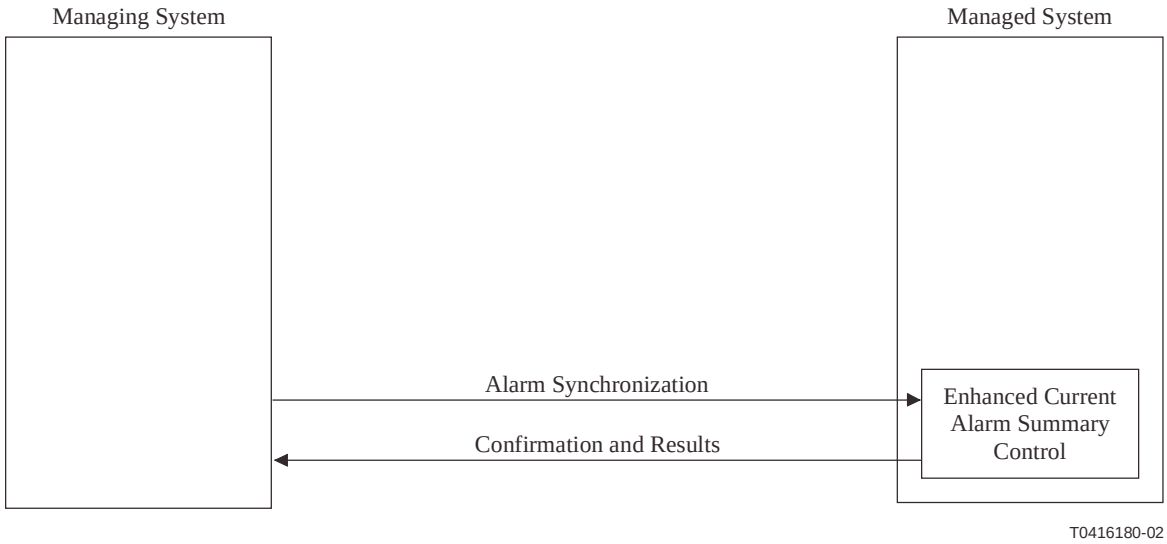

Figure II.16/Q.821.1 – Alarm Synchronization Functional Unit

II.2.16.1 Alarm Synchronization Service

The Alarm Synchronization service is used to request that an Enhanced Current Alarm Summary report be sent from the managed system to the managing system. It utilizes the alarmSynchronization method of the Enhanced Current Alarm Summary object. The alarm Synchronization method is described in 7.2.5.1.

The parameters for the Alarm Synchronization service are shown in the next subclauses.

II.2.16.1.1 Alarm Synchronization Parameters

Parameters for the Alarm Synchronization method in the Enhanced Current Alarm Summary Control object are as shown in Table II.17 (see ITU-T Rec. X.780 [18] for a more complete description of each parameter and ITU-T Rec. Q.816 [6] for a description of the OMG Notification Service structured events):

Table II.17/Q.821.1 – Alarm Synchronization Method Parameters

Parameter Name	Req/Ind	Rsp/Cnf	Notes
Name	O	–	Only used if access to the Enhanced Current Alarm Summary Control managed object is through a Facade object
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

Table II.17/Q.821.1 – Alarm Synchronization Method Parameters *(continued)*

Parameter Name	Req/Ind	Rsp/Cnf	Notes
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 [22]
Event Header	–	C	
Fixed Event Header	–	C	
Event Type	–	C	
Domain Name	–	C	
Type Name	–	C	
Event Name	–	C	
Optional Header Fields	–	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

Table II.17/Q.821.1 – Alarm Synchronization Method Parameters *(concluded)*

Parameter Name	Req/Ind	Rsp/Cnf	Notes
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	
Exceptions	–	P	

II.2.16.1.2 Get Next Parameters

Parameters for the Get Next method in the Alarm Synchronization Data Iterator object are as shown in Table II.18 (see ITU-T Rec. X.780 [18] for a more complete description of each parameter):

Table II.18/Q.821.1 – Get Next Method Parameters

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 [22]
Event Header	–	C	
Fixed Event Header	–	C	
Event Type	–	C	
Domain Name	–	C	
Type Name	–	C	
Event Name	–	C	
Optional Header Fields	–	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	

Table II.18/Q.821.1 – Get Next Method Parameters *(concluded)*

Parameter Name	Req/Ind	Rsp/Cnf	Notes
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	
Exceptions	–	P	

II.2.16.1.3 Destroy Parameters

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

II.3 Protocol Specification

II.3.1 Elements Of Procedure

Except for the services identified below, this Recommendation makes use of the elements of procedure defined for the services described in II.2.

II.3.1.1 Managed Objects

This Recommendation references the following support objects whose IDL is specified in OMG Notification Service [22]:

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 Recommendation references the following support objects whose IDL is specified in OMG Telecoms Log Service [23]:

a) Iterator;
b) Notify Log;

c) Notify Log Factory;

d) Typed Notify Log;

e) Typed Notify Log Factory.

This Recommendation references the following managed objects whose IDL are specified in ITU-T Rec. M.3120 [3]:

a) Alarm Severity Assignment Profile;

b) Managed Element;

c) Managed Element Complex;

d) Network.

This Recommendation references the following managed objects whose IDL are specified in ITU-T Rec. X.780 [18]:

a) Managed Object;

b) Managed Object Factory.

This Recommendation references the following support objects whose IDL are specified in this Recommendation (see clause 10):

a) Alarm Synchronization Data Iterator;

b) Current Alarm Summary Control;

c) Current Alarm Summary Control_F;

d) Current Alarm Summary Control Factory;

e) Enhanced Current Alarm Summary Control;

f) Enhanced Current Alarm Summary Control_F;

g) Enhanced Current Alarm Summary Control Factory;

h) Management Operations Schedule;

i) Management Operations Schedule_F;

j) Management Operations Schedule Factory.

This Recommendation references the following managed object whose IDL are specified in ITU-T Rec. X.780.1 [19]:

a) Managed Object_F.

II.3.1.2 Valuetypes

This Recommendation references the following valuetype specified in this Recommendation whose IDL are defined in ITU-T Rec. X.780 [18]:

a) Managed Object Value Type.

II.3.1.3 Types

This Recommendation references the following types associated with the objects specified in this Recommendation whose IDL are defined in ITU-T Rec. X.780 [18]:

a) AdministrativeStateType;

b) DeletePolicyType;

c) ExternalTimeType;

d) GeneralizedTimeType;

e) MONameType;

f) NameBindingType;

g) ObjectClassType;

h) OperationalStateType;

i) PerceivedSeverityType;

j) ProbableCauseType;

k) StartTimeType;

l) StopTimeType;

m) StringSetType;

n) TimePeriodType.

This Recommendation references the following types associated with the objects specified in this Recommendation whose IDL are defined in ITU-T Rec. Q.816 [6]:

a) FilterType;

b) LanguageType;

c) ScopeType.

This Recommendation references the following type associated with the objects specified in clause 6 whose IDL is defined in ITU-T Rec. M.3120 [3]:

a) AlarmStatusType.

This Recommendation references the following type associated with the objects specified in clause 6 whose IDL is defined in OMG Notification Service [22]:

a) EventBatch.

The objects defined in this Recommendation inherit attributes from ManagedObject as specified in ITU-T Rec. X.780 [18]; these attributes are not repeated here.

This Recommendation references the following types that are defined in 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) InvalidObjectInstanceErrorSeqType;

o) ObjectAlarmSummaryType;

p) ObjectListChoice;

q) ObjectListSetType;

r) ObjectListType;

s) ObjectOfReferenceType;

t) PerceivedSeveritySetType;
u) PerceivedSeverityTypeOpt;
v) ProbableCauseSetType;
w) ProbableCauseTypeOpt;
x) RangeOfObjectsType;
y) ScopedCriteriaSeqType;
z) ScopedCriteriaType;
aa) SimpleObjectListSetType;
bb) StartTimeType;
cc) StopTimeType;
dd) StopTimeType;
ee) SummaryContentsSetType;
ff) SummaryContentsType;
gg) TerminalRDNRangeType.

II.3.1.4 Notifications

This Recommendation references the following notifications defined in ITU-T Rec. X.780 [18]:

a) attributeValueChange;
b) objectCreation;
c) objectDeletion;
d) stateChange;
e) communicationsAlarm;
f) equipmentAlarm;
g) environmentalAlarm;
h) processingErrorAlarm;
i) qualityOfServiceAlarm.

This Recommendation references the following notification defined in ITU-T Rec. Q.816 [6]:

a) channelChange;
b) heartbeat.

This Recommendation references the following notification defined in this Recommendation:

a) currentAlarmSummaryReport.

II.3.1.5 Methods

This Recommendation references the following methods defined in ITU-T Rec. M.3120 [3]:

– In Managed Element:
  a) enableAudibleVisualLocalAlarmGet;
  b) enableAudibleVisualLocalAlarmSet;
  c) resetAudibleAlarm.

This Recommendation references the following methods defined in ITU-T Rec. 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 Recommendation are not repeated here.

This Recommendation references the following methods defined in this Recommendation.

– 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.

II.3.1.6 Exceptions

This Recommendation references the following exception specified in ITU-T Rec. M.3120 [3]:

a) NOAudibleVisualLocalAlarmPackage.

This Recommendation references the following exceptions specified in ITU-T Rec. X.780 [18]:

a) ApplicationError;

b) CreateError;

c) DeleteError.

This Recommendation references the following exceptions specified in ITU-T Rec. Q.816 [6]:

a) InvalidFilter;

b) InvalidParameter;

c) FilterComplexityLimit.

This Recommendation references the following exception specified in this Recommendation (see 10.4):

a) InvalidObjectInstanceErrorParameter;

b) NOmanagementOperationsScheduleOperationalStatePkg;

c) SelectionCriteriaNotSupported.

II.3.1.7 Name Bindings

This Recommendation references the following name bindings specified in this Recommendation (see 10.9):

a) CurrentAlarmSummaryControl_ManagedElement;

b) EnhancedCurrentAlarmSummaryControl_ManagedElement;

c) EnhancedCurrentAlarmSummaryControl_ManagedElementComplex;

d) EnhancedCurrentAlarmSummaryControl_Network;

e) ManagementOperationsSchedule_ManagedElement.

Appendix III

Changes from ITU-T Rec. Q.821

III.1 Remove Cancel Alarm Synchronization Action

The concept of Cancel Alarm Synchronization is no longer required. First, with CORBA, you can not 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 Recommendation, 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 Cancelled Alarm Synchronization Parameter parameters

III.2 No Longer Require Parameter Redefinition

In ITU-T Rec. Q.821 [8], 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 [11].

In ITU-T Rec. X.780 [18], 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 redefined.

This Recommendation removes the Suspect Object List Action Parameter.

III.3 Suspect Object List Moved To ITU-T Rec. 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 ITU-T Rec. Q.821 [8], but not used in ITU-T Rec. Q.821. The definition for this parameter with CORBA has been moved to ITU-T Rec. X.780 [18], where it is optionally included in alarms.

III.4 Changes To Alarm Info

In ITU-T Rec. X.780 [18], 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 the Notification Service. The container used for sets of notifications in the Notification Service is the OMG Notification Service [22] Event Batch. Note that Typed Events may be mapped so that they can be contained in an Event Batch (see 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.

III.5 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 ITU-T Rec. X.780 [18], 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.

III.6 Use Of Channels Instead Of EFD Associations

In Q/CMISE [9], Event Forwarding Discriminators (EFDs) contain associations for how to send notifications from an agent to a manager. In ITU-T Rec. Q.816 [6], the Notification Service 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 ITU-T Rec. Q.821 [8], contains the Destination Address attribute. It is used to allow the sending of the Current Alarm Summary Report notification to different associations. In ITU-T Rec. Q.816, 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.

SERIES OF ITU-T RECOMMENDATIONS

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

Geneva, 2002