T1M1

Internetwork Operations, Administration, Maintenance, and Provisioning

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

Accredited by the American National Standards Institute

Michael J. Fargano
Chairman

James T. Stanco
Vice Chairman

A Sponsored Committee of

Alliance for Telecommunications Industry Solutions

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

June 8, 2001

Albert H. Yuhan
3GPP TSG-SA/WG5 Chairman
albert.yuhan@voicestream.com

Subject: Response to 3GPP Liaison (S5-010217) on the Coordination of 3GPP and T1M1/ITU tML/XML Work Efforts

Dear Albert,

The following attachments are provided, for your consideration, in response to the 3GPP to T1M1 liaison - *S5-010217, April 6, 2001*:

- *T1M1.5/2001-096* - T1M1 Comments on 3GPP SA5 XML Based Performance Data Format and Comments on Bulk Configuration Data XML Format Specification (pages 2-3 of 99 in this liaison)
- *T1M1.5/2001-113* - Draft Rec. Q8.xx - CORBA-Based TMN Performance Management Service (pages 4-54 of 99 in this liaison)
- *T1M1.5/2001-081R1* - Proposal For A New Kind Of History Data Scanner (pages 55-59 of 99 in this liaison)
- *T1M1.5/2001-090R1* - Proposal for Using GDMO/ANS.1 Mappings to XML Schemas - Applied to Configuration Audit Support Function (X.792), (pages 60-99 of 99 in this liaison)

In addition to being attached, all individual documents are publicly available on our web site. Note that both T1M1.5/2001-081R1 and T1M1.5/2001-090R1 are being submitted to USSG B as proposed U S Contributions for consideration by ITU-T SG 4.

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

Best regards,

Mike Fargano
T1M1 Chairman

Cc (w/ Attachments):
Michael Truss, Vice Chair 3GPP TSG-SA WG5
Masaaki Yoshimura, Chair 3GPP2 TSG-S WG2
Jörg Schmidt, 3GPP2 to T1M1 Liaison
Geoff Caryer, 3GPP Work Item Contact
Athan Tuzel, Chair ATM Forum NM Working Group
Al Vincent, Technology Director TMF

Cc (w/o Attachments):
T1M1 Leadership Team
Tom Rutt - T1M1.5/2001-096 Contact
Weijing Chen - T1M1.5/2001-113 Contact
Dave Matthews - T1M1.5/2001-081R1 Contact
Raymond Reeves - T1M1.5/2001-090R1 Contact

COMMITTEE T1 – TELECOMMUNICATIONS
Committee T1M1
Greensboro NC; 14-18 May 2001

Contribution

TITLE: **Response to 3GPP liaison on the Co-ordination of 3GPP and T1M1/ITU tML/XML Work Efforts**

SOURCE*: **T1M1**

PROJECT: **tML**

REFs:

Doc.	Title	Source
T1M1/2001-43	Response to T1M1 liaison on the Co-ordination of 3GPP and T1M1/ITU tML/XML Work Efforts	3GPP SA5
T1M1/2001-42	File format for the transfer of bulk configuration data	3GPP SA5
T1M1/2001-40	3G Performance Management (PM) Release 4	3GPP SA5

ABSTRACT

T1M1 has reviewed 3GPP SA5 liaison documents 1m100410 and 1m100420** that illustrate the current use of XML within SA5. This liaisons contribution provides comments on these 3GPP SA5 documents.

**(T1M1 numbers corresponding to attachments noted in 3GPP liaison 1M100430)

*Tom Rutt terutt@lucent.com Tel: +1 732 949 7862

1. T1M1 Comments on 3GPP SA5 XML Based Performance Data Format

Your suggested format for XML Based performance data is within the scope of the Draft Rec Q.822Corba (the number is to be determined). This specification is based on Q.822, but is highly optimized for CORBA 2.3 (e.g., it uses value types for history data). It has history data records available from a "currentData" interface, related via containment based naming to the resource interface. There is also a history data scanner, which controls the generation of a PM data file, from data contained in history data records.

The draft output from the editors based on the Experts meeting in Pleasanton is attached. This is a delimited text file.

We suggest that an XML PM file format be presented at the Geneva SG4 meeting as a second format by 3GPP through an appropriate liaison partner.

The following comments on the detailed format of your File are provided for your consideration:

1. Your format does not seem to fit the typical XML principle of having meaningful names for the XML Tags. Your use of short hand cryptic tags will lead to difficulty in understanding the meaning of the performance data send via XML.

2. We cannot understand why you must send the source object name with each performance data value. Since each source object may have several parameters, this overhead of repeating the name, is against the principle of minimizing file size in common situations. Also, there appears to be no reason why the measurement type is sent in a separate sequence from the measurement data, since the measurement type could be included in the MeasurementData sequence. For example why not use a format like below:

   – PMData ::= SEQUENCE OF PMIntervalData
   – PMIntervalData ::= SEQUENCE { time TimeOfDay, duration TimeInterval, data Data}
   – Data::= SEQUENCE OF SEQUENCE {source ResourceName, measList MeasurementList}
   – MeasurementList ::= SEQUENCE OF SEQUENCE {measType MeasType, measValue MeasVal}

3. It is unclear whether your format supports multiple measurement intervals (granularity period for history data) in the same file. The attached draft shows that this feature is supported by the delimited text file format.

2. T1M1 Comments on Bulk Configuration Data XML format specification

We note that you discuss the contents of an XML file for bulk configuration data. However there is no discussion of mechanisms to request such a file or to deliver the results of such a request.

There is an implication that "modify" (e.g. create/delete/set) flag can be used in a "download" to change the MIB of the receiver. However, there is no explanation of transport mechanism (protocol) and trigger conditions for activation of the file transfer.

The Vendor Specific Data for a given MO class is validated by a vendor specific schema. It is unclear what advantage this has over having a "vendor Specific" schema extending the "standard" schema for the MO class using the extension phrase. The rationale for your approach versus the XML standard based extension mechanism needs to be clarified.

In your current file format, a separate schema is required for each MOC. However mOC type is an Enumeration. In order a new class will this imply adding to enumeration values and updating the version number. Using the subclass approach for vendor exceptions would require a more open set of MOC types, specific for each vendor. This might require eliminating the current use of ENUM to distinguish the MOC.

We are sending a contribution to the Geneva SG4 meeting on XML bulk configuration Audit file amendment for X.792. This is intended to have MO class specific schema, rather than relying on the vendor specific schema. Note that this is a working document and some open issues remain to be resolved (e.g. providing the managed object name).

We suggest that an XML Bulk config file format be presented at the Geneva SG4 meeting to help our efforts in creating an audit function using XML similar to X.792.

Attachments: Q.822Corba (Pleasanton output) T1M1.5/01-90R1 (file 1M150901.doc from T1M1 files - www.t1.org/t1m1/_m1-grid.htm)

COMMITTEE T1 – TELECOMMUNICATIONS **T1M1.5/2001-113**
Working Group: T1M1.5
Sub-Working Group: Management Services
Greensboro, NC; May 14 – 19, 2001

CONTRIBUTION

Title: Draft Rec. Q8.xx *"CORBA-based TMN Performance Management Service"*

Source: Editors

Contact: Weijing Chen
SBC Technology Resources
USA
Tel: +1 512 372 5710
Fax: +1 512 241 5710
E-mail: wchen@tri.sbc.com

Andy Mayer
Telcordia Technologies

Distribution: T1M1.5

Project: Protocol Standards for Communication between Operating Systems

---

ABSTRACT

This document defines an information model to be used in telecommunications performance management (PM) based on CORBA. It defines in Interface Definition Language (IDL) a set of interfaces, notifications, and constants. The intent of this document is to define a CORBA/IDL model similar to that defined in ITU Recommendations X.739 and Q.822 using CMISE. This document is compliant with proposed CORBA modeling standards Recommendation X.780, Q.816 and M.3120.

NOTICE

The proposals in this submission have been formulated to assist Standards committee T1M1. This document is offered to the committee as a basis for discussion and is not binding on SBC Communications and Lucent Technologies or any of its subsidiaries or affiliates. The requirements are subject to change in form and in numerical values after more study. SBC and Lucent specifically reserves the right to add to, or amend, the quantitative statements made herein. Nothing contained herein shall be construed as conferring by implication, estoppel, or otherwise any license or right under any patent, whether or not the use of information herein necessarily employs an invention of any existing or later issued patent.

ITU-Telecommunication Standardization Sector **Contribution _EGPL-19**

STUDY GROUP **4**

Geneva, Switzerland, 2 – 6 July, 2001

Questions: 14/4, 18/4, 19/4

Title: Draft Rec. Q8.xx *"CORBA-based TMN Performance Management Service"*

Source: Editors

Contact:

Weijing Chen
SBC Technology Resources
USA
Tel: +1 512 372 5710
Fax: +1 512 241 5710
E-mail: wchen@tri.sbc.com

Andy Mayer
Telcordia Technologies
USA

ABSTRACT

This document defines an information model to be used in telecommunications performance management (PM) based on CORBA. It defines in Interface Definition Language (IDL) a set of interfaces, notifications, and constants. The intent of this document is to define a CORBA/IDL model similar to that defined in ITU Recommendations X.739 and Q.822 using CMISE. This document is compliant with proposed CORBA modeling standards Recommendation X.780, Q816 and M.3120.

Acknowledgement: Bing Leng of Lucent Technologies contributed most original text of early version.

DRAFT
INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR
STUDY PERIOD 2001 - 2004

COM 4-xxx-E
July 2001
Original: English

Question: 19/4

STUDY GROUP 4 – CONTRIBUTION xxx

SOURCE*: EDITORs

TITLE: DRAFT NEW RECOMMENDATION Q.8xx: CORBA-based TMN Performance Management Service

Summary
This document defines an information model to be used in telecommunications performance management (PM) based on CORBA. It defines in Interface Definition Language (IDL) a set of interfaces, notifications, and constants. The intent of this document is to define a CORBA/IDL model similar to that defined in ITU Recommendations X.739 and Q.822 using CMISE. This document is compliant with proposed CORBA modeling standards Recommendation X.780, Q.816 and M.3120.

Source
ITU-T Recommendation Q.8xx was developed by ITU-T Study Group 4 (2001-2004) and was approved under the WTSC Resolution 1 procedure on the **xx of xx,** 2001.

Keywords
Common Object Request Broker Architecture (CORBA), Interface Definition Language (IDL), Managed Objects, Performance Management, TMN Interfaces

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

Foreword

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. The 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 Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 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

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The 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, the ITU had/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.

© ITU 2001

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the ITU.

Table of Contents

Foreword ..... **v**
Table of Contents ..... **vii**
Table of Figures ..... **ix**
1 Scope **1**
2 References **2**
3 Definitions **3**
4 Overview of the Performance Management Service **4**
4.1 REQUIREMENTS 4
4.1.1 Functional Requirements *4*
4.1.2 Design Requirements *5*
4.2 SERVICE MODELING 5
4.2.1 Scope *5*
4.2.2 UML Model *7*
4.2.3 Containment Relationship *10*
4.3 SERVICE INTERFACE DESCRIPTION 10
4.3.1 Scanner *10*
4.3.2 CurrentData *12*
4.3.2.1 CurrentData 12
4.3.2.2 HistoryData 14
4.3.3 ThreshodData *14*
4.3.4 HistoryDataFile (HDFile) *15*
4.3.5 HistoryDataScanner (HDScanner) *16*
5 Performance Management Service IDL **18**
5.1 IMPORTS 18
5.2 FORWARD DECLARATIONS 19
5.3 STRUCTURES AND TYPEDEFS 19
5.4 EXCEPTIONS 22
5.5 INTERFACES 23
5.5.1 Scanner *23*
5.5.2 CurrentData *24*
5.5.2.1 CurrentDataValueType 24
5.5.2.2 HistroyDataValueType 25
5.5.2.3 HistoryDataIterator 25
5.5.2.4 CurrentData 26
5.5.3 ThresholdData *28*
5.5.4 HDFile (HistoryDataFile) *29*
5.5.5 HDScanner (HistoryDataScanner) *30*
5.6 NOTIFICATIONS 32
5.7 NAME BINDING 32
5.7.1 ThresholdData *33*
5.7.2 HDFile *33*
5.7.3 HDScanner *33*
5.8 HDFILEFORMATCONST 34
Annex A (normative): File Format **35**

A. 1 Text File Format BNF 35
A. 2 Text File Format Description 37
Annex B (normative): CORBA File Format Convention 39
Annex C (informative): Example of CurrentDataValueType and HistoryDataValueType inheritance usage 39

Table of Figures

Figure 1. Use Cases for Performance Measurement 4
Figure 2. Class Diagram for CurrentData, HistoryData, and ThresholdData 7
Figure 3. Class Diagram for Scanner 8
Figure 4. Class Diagram for HistoryDataScanner and HistoryDataFile 9
Figure 5. Containment Relationship 10
Figure 6. PM File Format 38

Recommendation Q.8xx

CORBA-based TMN Performance Management Service (2001)

1 Scope

This document defines a support service to be used in telecommunications performance management (PM) based on CORBA. It defines in Interface Definition Language (IDL) a set of interfaces, notifications, and constants. The intent of this document is to define a CORBA/IDL specification similar to that defined in ITU Recommendations X.739 and Q.822 using CMISE. This document is compliant with proposed CORBA modeling standards Recommendation X.780, Q.816 and M.3120.

This document has the following sections:

Section 1.	Introduction
Section 2.	References
Section 3.	Definitions
Section 4.	Overview of the PM Service
Section 5.	PM Service IDL
Annex A.	File Format
Annex B.	Example of CurrentDataValueType and HsitoryDataValueType inheritance usage.

2 References

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

[1] The Object Management Group (OMG), "The Common Object Request Broker: Architecture and Specification", OMG Document formal/99-10-07, Revision 2.3.1, October 1999.

[2] The Object Management Group (OMG), "CORBA Services: Common Object Services Specification", Updated version, December 1998.

[3] The Object Management Group (OMG), "Notification Service", OMG TC Document telecom/98-11-01, November 3, 1998.

[4] ITU-T Recommendation, Q.816 (2001), "*CORBA-Based TMN Services*".

[5] ITU-T Recommendation, X.780 (2001), "*TMN Guidelines for Defining CORBA Managed Objects*".

[6] ITU-T Recommendation, M.3120 (2001), "*CORBA Generic Network and NE Level Information Model*".

[7] ITU-T Recommendation, X.739 (1993), "*Information technology – Open Systems Interconnection – Systems management: Metric objects and attributes*".

[8] ITU-T Recommendation, X.738 (1993), "*Information technology – Open Systems Interconnection – Systems management: Summarization function*".

[9] ITU-T Recommendation Q.822, (1994), "*Stage 1, Stage 2 and Stage 3 description for the Q3 interface – Performance management*".

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

[11] IEEE/ANSI Standard 1003.2 (1992) "*Information Technology – Portable Operating System Interface (POSIX) Part 2: Shell and Utilities*".

3 Definitions

The following terms and acronyms are defined in Recommendation X.739, X.780 or Q.822 and used in this document.

DN	Distinguished Name
NE	Network Element
PM	Performance Management
QoS	Quality of Service
TMN	Telecommunications Management Network.

4 Overview of the Performance Management Service

4.1 Requirements

4.1.1 Functional Requirements

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

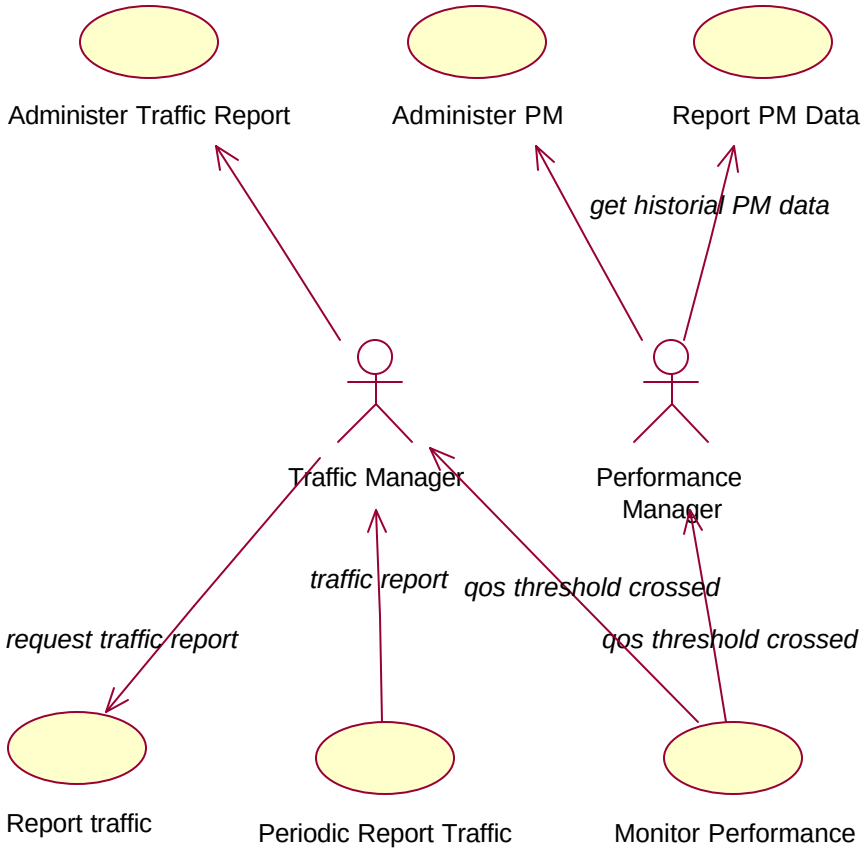


Figure 1. Use Cases for Performance Measurement

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

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

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

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

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

4.1.2 Design Requirements

This document follows the information modeling guidelines defined in ITU-T Rec. X.780. In addition, this document tries to keep the model practical, simple and make sense to the users and products.

4.2 Service Modeling

4.2.1 Scope

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

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

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

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

For this purpose, HistoryDataScanner is designed to retrieve history data saved under the relevant CurrentData objects. A file transfer mechanism is used in HistoryDataScanner for retrieving large amount of data since other techniques are likely to be both slower and to consume more computing and communication resources. The recommend supports the ability to control the generation of the file, the selection of the format of the file, and the notification of the user of when the file is produced, along with such information as is needed to obtain the file. However, specific file transfer mechanism and file administration are outside the scope of the document.

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

4.2.2 UML Model

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

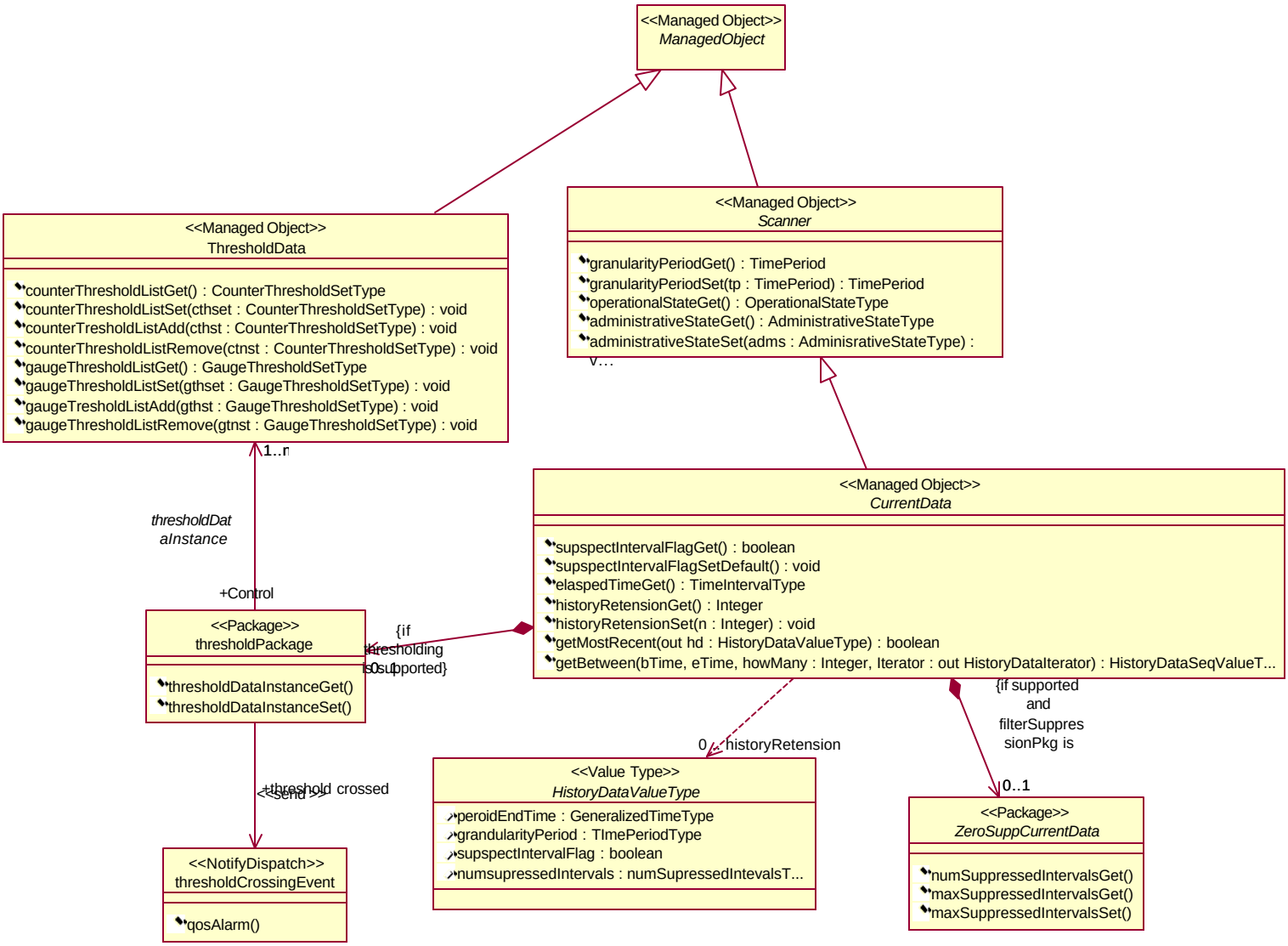


Figure 2. Class Diagram for CurrentData, HistoryData, and ThresholdData

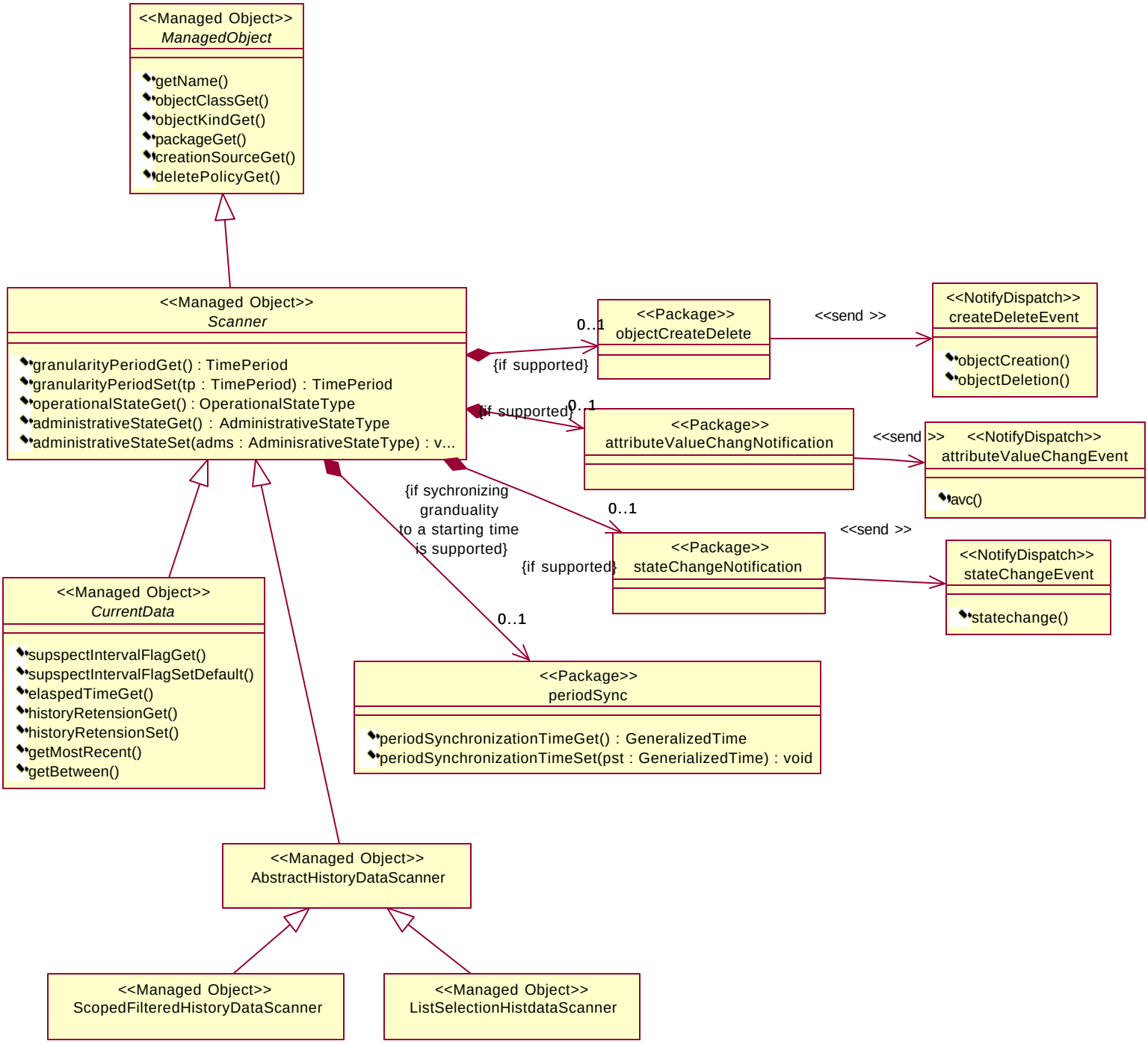


Figure 3. Class Diagram for Scanner

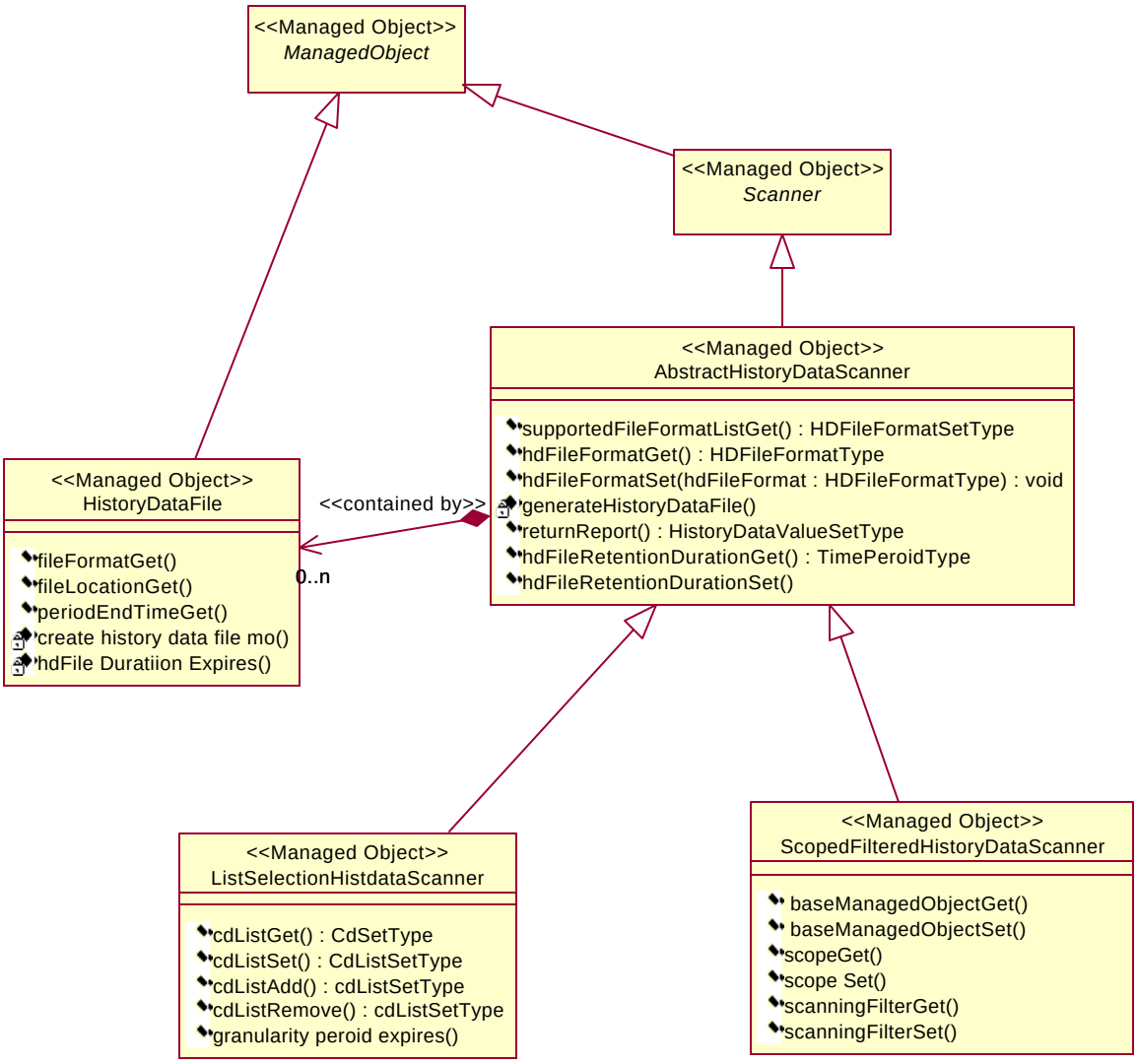


Figure 4. Class Diagram for HistoryDataScanner and HistoryDataFile

4.2.3 Containment Relationship

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

[Editor note: to be consistent with file format, Network to AbstractHistoryDataScanner shall be added.]

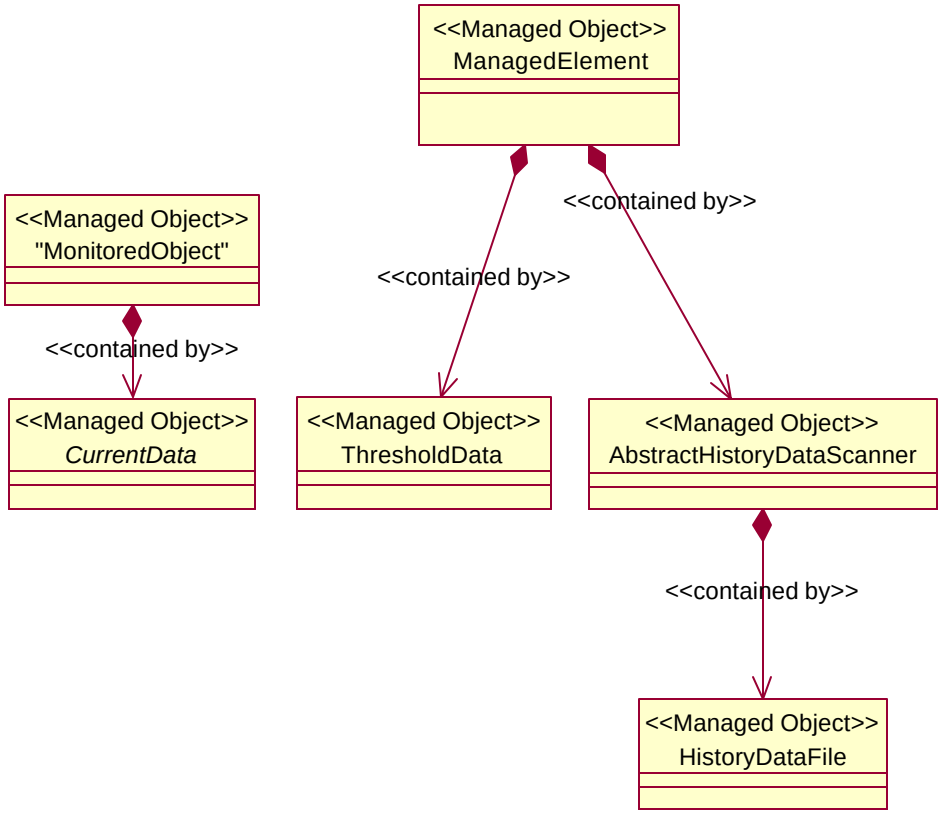


Figure 5. Containment Relationship

4.3 Service Interface Description

4.3.1 Scanner

A managed object of this interface represents the ability to retrieve values of attributes of managed objects and produce summary information from those values. This summary information may be made available in attributes, notifications, action replies, or some

combination of these. Summary information may consist of observed attribute values or statistics calculated from these values (either over time or over managed objects).

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

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

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

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

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

The following attributes are defined in the entity:

- *administrativeState* – is used to activate and deactivate (administratively suspend and resume) the function performed by the scanner.
- *granularityPeriod* – specifies the length of time during which the "scan" is performed.
- *operationalState* – identifies whether or not the scanning function represented by this object is capable of performing its normal functions.

The periodSynchronizationPackage is present if configurable agent internal synchronization of repeating time periods is required.

The following attributes are defined in the periodSynchronizationPackage:

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

The Scanner interface is not instantiable.

4.3.2 CurrentData

4.3.2.1 CurrentData

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

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

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

- getMostRecent – will retrieve the most recent history data. The "most recent" means the most recent history data record saved, which may be several intervals before the current time because the historyRetention may be set to zero or the measurements collected are all zero. It is possible the getMostRecent may not be able to retrieve any data back. For example, all measurements are zeros (no history data is saved) or historyRetention has been set to zero. The boolean value returned by getMostRecent method indicates the availability of the data. True means the out parameter contains the data, false the parameter is undefined.
- getBetween – will retrieve a set of history data within a time window specified by start and end time parameters. The resultIterator is used to transfer large chunk of data. If howMany is not provided, then the initial amount of data returned is a local matter.

The following attributes are defined in the entity:

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

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

If thresholdPackage is present, the CurrentData object contains at least a pointer to a ThresholdData object. If any of the thresholds (defined in the referenced ThresholdData object) are violated a Quality of Service (QoS) alarm notification is emitted by the CurrentData object. The thresholdInfo field of the alarm shall contain the measurement attribute which violates the threshold. If additional threshold crossing(s) occur during the current time interval then additional alarms shall be emitted as well.

The thresholdDataInstanceList attribute contains a set of pointers to ThresholdData objects, specified by managing system(s). Therefore, more than one QoS Alarm notifications might be emitted for a single monitored attribute. And it is possible the notifications might be conflict. For example, one notification indicates alarm is on, the other cleared. It is the managing systems' responsibility to maintain the notification consistency.

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

The following attributes are defined in the thresholdPackage:

- *thresholdDataInstanceList* – is a "list" attribute in which each item is a pointer to a ThresholdData object that contains threshold limits for performance parameters.

zeroSuppressionPackage is a conditional package of CurrentData. When this package is present and an interval terminates with 'all-zeros' performance measurements, there is no history data structure is created.

The following attributes are defined in the zeroSuppressionPackage:

- *numSuppressedIntervals* – is used to count the number of consecutive intervals for which suppression (i.e., non-creation of a history data structure) has occurred. This attribute reflects performance measurements up to, but not including, the current interval. This attribute gets incremented at the end of an interval if suppression has occurred. Otherwise, the attribute is reset. Or it reaches maxSuppressedIntervals, a history data record will be created and itself will be reset.
- *maxSuppressedIntervals* – limits the maximum number of suppressed intervals that will be collected without creating a structure of the history data. The default value -1

means that no limit on the number of consecutive suppressed intervals. On the other hand, the maxSuppressedIntervals is effectively equal to infinity.

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

The CurrentData interface is not instantiable.

4.3.2.2 HistoryData

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

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

The following attributes are defined in the entity:

- *periodEndTime* – indicates the time at the end of the interval.
- *granularityPeriod* – is used to copy the same attribute from the CurrentData object.
- *suspectIntervalFlag* – is used to copy the same attribute from the CurrentData object.
- *numSupressedIntevals* – indicates the number of suppressed intervals has occurred before the creation of this history valuetype record. It has default value 0. When copying from CurrentData object, the default value is used. When copying from CurrentData with ZeroSuppressionPackage, the corresponding attribute value is used.

4.3.3 ThreshodData

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

Thresholds are established by use of the managed object class ThresholdData. The ThresholdData objects specify the thresholds being applied. Threshold values in a ThresholdData object may apply to multiple CurrentData objects and ThresholdData objects are pointed to by CurrentData objects. The CurrentData, or its subclasses, object

indicates the collection interval used with the threshold. A qualityOfServiceAlarm notification is generated whenever a threshold is crossed.

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

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

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

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

The counterThresholdListPackage is present if an instance supports it and the gaugeThresholdListPackage is not present.

The following attributes are defined in the counterThresholdListPackage:

- *counterThresholdList* – contains a set of threshold settings for performance attributes of the counter type (e.g., erred seconds). Each threshold setting consists of the attribute identifier, the threshold value and (optionally) the severity of the threshold-exceeded event.

The gaugeThresholdListPackage is present if an instance supports it and the counterThresholdListPackage is not present

The following attributes are defined in the gaugeThresholdListPackage:

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

4.3.4 HistoryDataFile (HDFile)

HistoryDataFile is a subclass of ManagedObject. It one-to-one maps to history data file. Each history data file has one associated HistoryDataFile object. Each HistoryDataFile object has one underlying history data file. The HistoryDataFile object is created only after history data file is produced. The HistoryDataFile object is deleted immediately after history data file is purged, and vice versa that history data file is purged if HistoryDataFile object is deleted.

The HistoryDataFile object has the following attributes:

- fileLocation – identifies file location in URL format
- fieFormat – indicates which file format used.
- PeriodEndTime - indicates the time at the end of the file generation.

When a history data file is generated and a HistoryDataFile object is created, an objectCreation notification will be sent from managed system to inform managing system the availability of the history data file. When a history data file is removed and the associated HistoryDataFile object is deleted, an objectDeletion notification could be sent from managed system to managing system if objectDeleteNotificationPackage is supported.

4.3.5 AbstractHistoryDataScanner (AbstractHDScanner)

HistoryDataScanner is a subclass of Scanner. It supports scanning and reporting history data by providing the ability to control the generation of history data file, the selection of the format of the file, and the notification of user when the file is produced.

To generate files, the managing system specifies the request through a set of attributes:

- *hdFileFormat* - indicates which file format to use
- *currentDataList* - contains a set of current data objects whose historical data needs to be scanned/reported to a file
- *begin relative time* - the begin time relative to the current time together with the end relative time defining a window to select history data. The negative value of begin relative time means past time.
- *end relative time* - the end time relative to the current time. The negative value of end relative time means past time. The zero value means no ending time which would result periodically file generation until stopped by managing system.
- *granularityPeirod* - sets a time window so that the history data with their time stamps fall into the window will be reported within a file.

Before setting values to these attributes, the administrativeState has to be set to 'lock'. And when the administrativeState is set to 'unlock', the periodically file generation begins.

At the end of each granularity period, the HistoryDataScanner scans the historical data from a set of current data objects, specified in currentDataList, collects any data if its time stamp falls within this current granularity time interval, and generates a data file in a format indicated in hdFileFormat. When the file is produced, a HistoryDataFile object is created by managed system and an objectCreation notification is emitted to the managing system. The source of objectCreation notification is the just created HistoryDataFile object.

If the granularity period is zero, then there is no periodic file generation.

The managed system may support more than one file format. The attribute supportedFileFormatList contains those supported file formats.

4.3.6 ListSelectionHistoryDataScanner

[Editor's Note: place holder]

4.3.7 ScopedFilteredHistoryDataScanner

[Editor's Note: place holder]

5 Performance Management Service IDL

```
#ifndef _itut_q8xx_idl_
#define _itut_q8xx_idl_


#include <itut_x780.idl>


#pragma prefix "itu.int"


/**
This IDL code is intended to be stored in a file named "itut_q8xx.idl"
located in the search path used by IDL compilers on your system.
*/




/**
This module, itut_q8xx, contains IDL definition based on objects defined in
X.739 and Q.822.  The IDL definitions in this file are the object interfaces.
*/
module itut_q8xx
{




/**
```

5.1 Imports

```
*/

/**
Types imported from itut_x780
*/
        typedef itut_x780::AdministrativeStateType AdministrativeStateType;
        typedef itut_x780::GeneralizedTimeType GeneralizedTimeType;
        typedef itut_x780::IstringType IstringType;
        typedef itut_x780::IstringSetType IstringSetType;
        typedef itut_x780::MONameType MONameType;
        typedef itut_x780::MONameSetType MONameSetType;
        typedef itut_x780::NameBindingType NameBindingType;
        typedef itut_x780::OperationalStateType OperationalStateType;
        typedef itut_x780::PerceivedSeverityType PerceivedSeverityType;
        typedef itut_x780::UIDType UIDType;


/**
Interfaces imported from itut_x780

itut_x780::ManagedObject
itut_x780::ManagedObjectFactory
*/
```

```
/**
```

5.2 Forward Declarations

```
*/

/**
Interface forward declarations
*/
        interface Scanner;
        interface HistoryDataIterator;
        interface CurrentData;
        interface ZeroSuppCurrentData;
        interface ThresholdData;
        interface HDFile;
        interface AbstractHDScanner;
        interface ListSelectionHDScanner;
        interface ScopedFiltedHDScanner;


/**
Valuetype forward declarations
*/
        valuetype ScannerValueType;
        valuetype CurrentDataValueType;
        valuetype HistroyDataValueType;
        valuetype ZeroSuppCurrentDataValueType;
        valuetype ThresholdDataValueType;
        valuetype HDFile;
        valuetype AbstractHDScannerValueType;
        valuetype ListSelectionHDScanner;
        valuetype ScopedFiltedHDScanner;


/**
Interface forward declarations
*/
        typedef MONameType ScannerNameType;
        typedef MONameType CurrentDataNameType;
        typedef MONameType ZeroSuppCurrentDataNameType;
        typedef MONameType ThresholdDataNameType;
        typedef MONameType HDFileNameType;
        typedef MONameType AbstractHDScannerNameType;
        typedef MONameType ListSeletionHDScannerNameType;
        typedef MONameType ScopedFiltedHDScannerNameType;




/**
```

5.3 Structures and Typedefs

```
*/

/**
This data type defines a sequence of HistroyDataValueType. The order is
significant.
*/
        typedef sequence <HistroyDataValueType> HistoryDataSeqValueType;

        typedef sequence <CurrentDataNameType> CurrentDataNameSetType;
```

```
        typedef UIDType HDFileFormatType;

        typedef sequence <HDFileFormatType> HDFileFormatSetType;

        typedef IstringType URLType;

/**
PerceivedSeverityTypeOpt is an optional type. If the discriminator is true
the value is present, otherwise the value is null.
*/
        union PerceivedSeverityTypeOpt switch (boolean)
        {
                case TRUE:
                        PerceivedSeverityType   val;
        };

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

Editor Note: A clarification is needed in X.739 section 8.1.9.4.3 on the
behaviour of the severity indicating gauge threshold. Currently it is unclear
what value of PerceivedSeverity is sent in the threshold crossing notification
PerceivedSeverity field if notify is set on but the optional PerceivedSeverity
value of the threshold is not present in the severity indication gauge
threshold syntax. We suggest that in this case the indeterminate value be sent
unless the managed object behaviour specifies a different value. The following
text should be added to the end of this section:

If a notify-high or notify-low switch is on (true) and the severity indication
value is not present, then the indeterminate value is sent in the notification
unless the managed object behaviour specifies another value.
*/
        struct SeverityIndicatingThresholdType
        {
                float                           threshold;
                boolean                         notifyOnOff;
                PerceivedSeverityTypeOpt        severityIndication;
        };

        struct CounterThresholdSettingType
        {
                string                          attributeName;
                SeverityIndicatingThresholdType threshold;
        };

/**
The order is insignificant.
*/
        typedef sequence <CounterThresholdSettingType> CounterThresholdSetType;

/**
The attribute of SeverityIndicatingGaugeThresholdType has similar behaviour to
the gauge-threshold attribute defined in X.721. The syntax has an added
parameter for indicating the associated severity, as defined in
SeverityIndicatingThresholdType, to the notification triggered by the crossing
of the corresponding threshold level. As an enhancement to the syntax of the
```

```
gauge-threshold attribute type it adds an optional severity indication
parameter to the syntax of both the notify-high and notify-low sub-members
within each threshold level member. This attribute type has additional
behaviour associated with these optional perceived severity indication
parameters, which is defined as follows:

- If the notify-high's switch is on (true), the notify-high's severity
indication value shall be reported in the perceived severity parameter of a
notification triggered by the gauge value crossing the notify-high's
gauge-threshold value in the positive going direction.

- If the notify-low's switch is on (true), the notify-low's severity
indication value shall be reported in the perceived severity parameter of a
notification triggered by the gauge value crossing the notify-low's
gauge-threshold value in the negative going direction.

- If both switches are on (true) for a single threshold level, one of the
severity indication values shall be "clear". The severity indicating
gauge-threshold shall only emit a clear event notification if the
corresponding threshold level (either notify-high or notify-low) notification
has been emitted and no other clear notification for this threshold level pair
has been emitted since the previous corresponding threshold level notification
has been emitted.
*/
        struct SeverityIndicatingGaugeThresholdType
        {
                SeverityIndicatingThresholdType notifyLow;
                SeverityIndicatingThresholdType notifyHigh;
        };

/**
The order is insignificant.
*/
        typedef sequence<SeverityIndicatingGaugeThresholdType>
                SeverityIndicatingGaugeThresholdSetType;

        struct GaugeThresholdSettingType
        {
                string                                  attributeName;
                SeverityIndicatingGaugeThresholdSetType threshold;
        };

/**
The order is insignificant.
*/
        typedef sequence<GaugeThresholdSettingType>
                GaugeThresholdSetType;

        struct TimeIntervalType
        {
                unsigned short day;
                unsigned short hour;
                unsigned short minute;
                unsigned short second;
                unsigned short ms;
        };

/**
The following definitions are translated from X.739 ASN.1 definitions.
*/
        enum TimePeriodChoice
        {
                timePeriodChoiceDays,
```

```
                timePeriodChoiceHours,
                timePeriodChoiceMinutes,
                timePeriodChoiceSeconds,
                timePeriodChoiceMilliSeconds,
                timePeriodChoiceMicroSeconds,
                timePeriodChoiceNanoSeconds,
                timePeriodChoicePicoSeconds
        };

        union TimePeriodType switch (TimePeriodChoice)
        {
                case timePeriodChoiceDays:
                        long day;
                case timePeriodChoiceHours:
                        long hour;
                case timePeriodChoiceMinutes:
                        long minute;
                case timePeriodChoiceSeconds:
                        long second;
                case timePeriodChoiceMilliSeconds:
                        long ms;
                case timePeriodChoiceMicroSeconds:
                        long us;
                case timePeriodChoiceNanoSeconds:
                        long ns;
                case timePeriodChoicePicoSeconds:
                        long ps;
        };



/**
```

5.4 Exceptions

```
*/

/**
Exceptions and Constants for Conditional Package
*/
        exception NOcounterThresholdListPackage {};

        exception NOdeleteNotificationPackage {};

        exception NOgaugeThresholdListPackage {};

        exception NOperiodSynchronizationPackage {};

        exception NOthresholdPackage {};


        const string counterThresholdListPackage =
                "itut_q8xx::counterThresholdListPackage";
        const string deleteNotificationPackage =
                "itut_q8xx::deleteNotificationPackage";
        const string gaugeThresholdListPackage =
                "itut_q8xx::gaugeThresholdListPackage";
        const string periodSynchronizationPackage =
                "itut_q8xx::periodSynchronizationPackage";
        const string thresholdPackage =
                "itut_q8xx::thresholdPackage";
```

```
/**
Exceptions for Performance Management
*/




/**
```

5.5 Interfaces

```
*/

/**
```

5.5.1 Scanner

```
See Section 4.3.1 for behaviour.

This interface contains the following CONDITIONAL PACKAGES.

- periodSynchronizationPackage: presnet if configurable agent internal
synchronization of repeating time periods is required.

The Scanner interface is not instantiable.
*/
        valuetype ScannerValueType: itut_x780::ManagedObjectValueType
        {
                public AdministrativeStateType  administrativeState;
                        // GET-REPLACE
                public OperationalStateType     operationalState;
                        // GET
                public TimePeriodType           granularityPeriod;
                        // GET-REPLACE
                public GeneralizedTimeType      periodSynchronizationTime;
                        // conditional
                        // periodSynchronizationPackage
                        // GET-REPLACE

        }; // valuetype ScannerValueType

        interface Scanner: ManagedObject
        {
/**
administrativeState is a read/write attribute supported by Get and Set access
operations.
/*
                AdministrativeStateType administrativeStateGet ()
                        raises (itut_x780::ApplicationError);

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

/**
The attribute operationalState is read-only supported by Get access operation.
*/
                OperationalStateType operationalStateGet ()
                        raises (itut_x780::ApplicationError);
/**
granularityPeriod is a read/write attribute supported by Get and Set access
```

```
operations.
*/
                TimePeriodType granularityPeriodGet ()
                        raises (itut_x780::ApplicationError);

                void granularityPeriodSet
                        (in TimePeriodType granularityPeriod)
                        raises (itut_x780::ApplicationError);

/**
periodSynchronizationTime is a read/write attribute supported by Get and Set
access operations.
*/
                GeneralizedTimeType periodSynchronizationTimeGet ()
                        raises (itut_x780::ApplicationError,
                                NOperiodSynchronizationPackage);

                void periodSynchronizationTimeSet
                        (in GeneralizedTimeType periodSyncTime)
                        raises (itut_x780::ApplicationError,
                                NOperiodSynchronizationPackage);

                CONDITIONAL_NOTIFICATION
                        (itut_x780::Notifications, objectCreation,
                        createDeleteNotificationsPackage)
                CONDITIONAL_NOTIFICATION
                        (itut_x780::Notifications, objectDeletion,
                        createDeleteNotificationsPackage)
                CONDITIONAL_NOTIFICATION
                        (itut_x780::Notifications, attributeValueChange,
                        attributeValueChangeNotificationPackage)
                CONDITIONAL_NOTIFICATION
                        (itut_x780::Notifications, stateChange,
                        stateChangeNotificationPackage)

        };  // interface Scanner



/**
```

5.5.2 CurrentData

```
See section 4.3.2.1 for behaviour.

This interface contains the following CONDITIONAL PACKAGES.

- thresholdPackage: present if a Quality of Service Alarm Notification is to
be emitted for threshold crossing.

- zeroSupressionPackage: present if an instance supports it.

The CurrentData interface is not instantiable.
*/

/**
```

5.5.2.1 CurrentDataValueType

```
*/
        valuetype CurrentDataValueType: ScannerValueType
```

```
        {
                public boolean                  suspectIntervalFlag;
                        // GET REPLACE-WITH-DEFAULT
                public TimeIntervalType         elapsedTime;
                        // GET
                public short                    historyRetention;
                        // Mandatory now (used to be conditional)
                        // GET-REPLACE
                public MONameSetType            thresholdDataInstanceList;
                        // conditional
                        // thresholdPackage
                        // GET-REPLACE ADD-REMOVE
                public short                    numSuppressedIntervals;
                        // conditional
                        // zeroSuppressionPackage
                        // GET
                public short                    maxSuppressedIntervals;
                        // conditional
                        // zeroSuppressionPackage
                        // GET-REPLACE
        }; // valuetype CurrentDataValueType


/**
```

5.5.2.2 HistroyDataValueType

```
See section 4.3.2.2 for behaviour.
*/
        valuetype HistroyDataValueType
        {
                public GeneralizedTimeType      periodEndTime;
                        // GET
                public TimePeriodType           granularityPeriod;
                        // GET
                public boolean                  suspectIntervalFlag;
                        // GET
                public short                    numSupressedIntevals;
                        // GET
        }; // valuetype HistroyDataValueType


/**
```

5.5.2.3 HistoryDataIterator

```
HistoryDataIterator is used for iteratively transferring large chunk of
history data.
*/
        interface HistoryDataIterator
        {
/**
This method is used to return the next how many history data. howMany is the
maximum number of history data for which results should be return in first
batch, must be non-zero. historyDataList is a sequence of history data records.
The method will return true if there is data in out parameter, false otherwise.
*/
                boolean getNext
                        (in unsigned short howMany,
                        out HistoryDataSeqValueType historyDataList)
                        raises (itut_x780::ApplicationError);
```

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

        };  // interface HistoryDataIterator


/**
```

5.5.2.4 CurrentData

```
*/
        interface CurrentData: Scanner
        {
/**
Define default value for suspectIntervalFlag
*/
                const boolean suspectIntervalFlagDefault = FALSE;

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

/**
This attribute is read-only, supported by Get access operation, however it can
be replaced by default, which is supported by DefaultSet operation.
*/
                boolean suspectIntervalFlagGet ()
                        raises (itut_x780::ApplicationError);

                void suspectIntervalFlagDefaultSet
                        (in boolean suspectIntervalFlag)
                        raises (itut_x780::ApplicationError);

/**
The attribute is read-only, supported by Get access operation.
*/
                TimeIntervalType elapsedTimeGet ()
                        raises (itut_x780::ApplicationError);

/**
This attribute is read/write, supported by Get and Set access operations.
*/
                short historyRetentionGet ()
                        raises (itut_x780::ApplicationError);

                void historyRetentionSet
                        (in short intervalNum)
                        raises (itut_x780::ApplicationError);

/**
This attribute is supported by Get, Set, Add, and Remove access operations.
*/
                MONameSetType thresholdDataInstanceListGet ()
                        raises (itut_x780::ApplicationError,
                                NOthresholdPackage);
```

```
                void thresholdDataInstanceListSet
                        (in MONameSetType dataInstanceList)
                        raises (itut_x780::ApplicationError,
                                NOthresholdPackage);

                void thresholdDataInstanceListAdd
                        (in MONameSetType dataInstanceList)
                        raises (itut_x780::ApplicationError,
                                NOthresholdPackage);

                void thresholdDataInstanceListRemove
                        (in MONameSetType dataInstanceList)
                        raises (itut_x780::ApplicationError,
                                NOthresholdPackage);

/**
This method is used to retrieve most recent history data record.
*/
                boolean getMostRecent
                        (out HistroyDataValueType historyData)
                        raises (itut_x780::ApplicationError);

/**
The getBetween method will retrieve a sequence of history data records falling
within the time window specified by the startTime and endTime parameters.
*/
                HistoryDataValueSeqType getBetween
                        (in GeneralizedTimeType startTime,
                        in GeneralizedTimeType endTime,
                        in unsigned short howMany,
                        out HistoryDataIterator resultIterator)
                        raises (itut_x780::ApplicationError);

/**
The attribute is read-only, supported by Get access operations.
Editor Note: it was read/write in Q.822.
*/
                short numSuppressedIntervalsGet ()
                        raises (itut_x780::ApplicationError);

/**
This attribute is read/write, supported by Get and Set access operations.
*/
                short maxSuppressedIntervalsGet ()
                        raises (itut_x780::ApplicationError);

                void maxSuppressedIntervalsSet
                        (in short maxSuppInterval)
                        raises (itut_x780::ApplicationError);

                CONDITIONAL_NOTIFICATION
                        (itut_x780::Notifications, qualityOfServiceAlarm,
                        NOthresholdPackage)

        };  // interface CurrentData



/**
```

5.5.3 ThresholdData

```
See section 4.3.3 for behaviour.

This interface contains the following CONDITIONAL PACKAGES.

- counterThresholdListPackage: present if an instance supports it and the
gaugeThresholdListPackage is not present.

- gaugeThresholdListPackage: present if an instance supports it and the
counterThresholdListPackage is not present
*/
        valuetype ThresholdDataValueType: itut_x780::ManagedObjectValueType
        {
                public CounterThresholdSetType  counterThresholdList;
                        // conditional
                        // counterThresholdListPackage
                        // GET-REPLACE ADD-REMOVE
                public GaugeThresholdSetType    gaugeThresholdList;
                        // conditional
                        // gaugeThresholdListPackage
                        // GET-REPLACE ADD-REMOVE
        }; // valuetype ThresholdDataValueType


        interface ThresholdData: ManagedObject
        {
/**
The attribute is supported by Get, Set, Add, and Remove access operations.
*/
                CounterThresholdSetType counterThresholdListGet ()
                        raises (itut_x780::ApplicationError,
                                NOcounterThresholdListPackage);

                void counterThresholdListSet
                        (in CounterThresholdSetType counterThresholdList)
                        raises (itut_x780::ApplicationError,
                                NOcounterThresholdListPackage);

                void counterThresholdListAdd
                        (in CounterThresholdSetType counterThresholdList)
                        raises (itut_x780::ApplicationError,
                                NOcounterThresholdListPackage);

                void counterThresholdListRemove
                        (in CounterThresholdSetType counterThresholdList)
                        raises (itut_x780::ApplicationError,
                                NOcounterThresholdListPackage);

/**
The attribute is supported by Get, Set, Add, and Remove access operations.
*/
                GaugeThresholdSetType gaugeThresholdListGet ()
                        raises (itut_x780::ApplicationError,
                                NOgaugeThresholdListPackage);

                void gaugeThresholdListSet
                        (in GaugeThresholdSetType gaugeThresholdList)
                        raises (itut_x780::ApplicationError,
                                NOgaugeThresholdListPackage);

                void gaugeThresholdListAdd
```

```
                        (in GaugeThresholdSetType gaugeThresholdList)
                        raises (itut_x780::ApplicationError,
                                NOgaugeThresholdListPackage);

                void gaugeThresholdListRemove
                        (in GaugeThresholdSetType gaugeThresholdList)
                        raises (itut_x780::ApplicationError,
                                NOgaugeThresholdListPackage);

                CONDITIONAL_NOTIFICATION
                        (itut_x780::Notifications, objectCreation,
                        createDeleteNotificationsPackage)
                CONDITIONAL_NOTIFICATION
                        (itut_x780::Notifications, objectDeletion,
                        createDeleteNotificationsPackage)
                CONDITIONAL_NOTIFICATION
                        (itut_x780::Notifications, attributeValueChange,
                        attributeValueChangeNotificationPackage)

        };  // interface ThresholdData


        interface ThresholdDataFactory: ManagedObjectFactory
        {
                ThresholdData create
                        (in NameBindingType nameBinding,
                        in MONameType superior,
                        inout Istring name,     // auto naming if null
                        in IstringSetType packageNameList,
                        in CounterThresholdSetType counterThresholdList,
                                // conditional
                                // counterThresholdListPackage
                                // GET-REPLACE ADD-REMOVE
                        in GaugeThresholdSetType gaugeThresholdList)
                                // conditional
                                // gaugeThresholdListPackage
                                // GET-REPLACE ADD-REMOVE
                        raises (itut_x780::ApplicationError,
                                itut_x780::CreateError);

        }; // interface ThresholdDataFactory




/**
```

5.5.4 HDFile (HistoryDataFile)

```
See section 4.3.4 for behaviour.

This interface contains the following CONDITIONAL PACKAGES.

- objectDeleteNotificationPackage: present if an instance supports it.
*/
        valuetype HDFileValueType: ManagedObjectValueType
        {
                public HDFileFormatType         fileFormat;
                        // GET, SET-BY-CREATE
                public URLType                  fileLocation;
                        // GET, SET-BY-CTEATE
```

```
                public GeneralizedTimeType      periodEndTime;
                        // GET, SET-BY-CREATE
                public TimePeriodType           fileRetentionDuration;
                        // GET, SET-BY-CREATE
        }; // valuetype HDFileValueType


        interface HDFile: itut_x780::ManagedObject
        {
                HDFileFormatType fileFormatGet ()
                        raises (itut_x780::ApplicationError);

                URLType fileLocationGet ()
                        raises (itut_x780::ApplicationError);

                GeneralizedTimeType periodEndTimeGet ()
                        raises (itut_x780::ApplicationError);

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

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

        }; // interface HDFile


        interface HDFileFactory: itut_x780::ManagedObjectFactory
        {
                HDFile create
                        (in NameBindingType nameBinding,
                        in MONameType superior,
                        inout Istring name, // auto naming if empty string
                        in HDFileFormatType fileFormat,
                        in URLType fileLocation,
                        in GeneralizedTimeType periodEndTime,
                        in TimePeriodType fileRetentionDuration)
                        raises (itut_x780::ApplicationError,
                                itut_x780::CreateError);

        }; // interface HDFileFactory



/**
```

5.5.5 AbstractHDScanner (AbstractHistoryDataScanner)

```
See section 4.3.5 for behaviour.
*/
        valuetype AbstractHDScannerValueType: ScannerValueType
        {
                public HDFileFormatSetType      supportedFileFormatList;
                        // GET
                public HDFileFormatType         hdFileFormat;
                        // GET, SET
                public TimePeriodType           hdFileRetentionDuration;
                        // GET, SET
```

```
        }; // valuetype HDScannerValueType


        interface AbstractHDScanner: Scanner
        {

                HDFileFormatSetType supportedFileFormatListGet ()
                        raises (itut_x780::ApplicationError);

/**
The hdFileFormat must be one of supported file format.
*/
                HDFileFormatType hdFileFormatGet ()
                        raises (itut_x780::ApplicationError);

                void hdFileFormatSet
                        (in HDFileFormatType hdFileFormat)
                        raises (itut_x780::ApplicationError);

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

                void hdFileRetentionDurationSet
                        (in TimePeriodType hdFileRetentionDuration)
                        raises (itut_x780::ApplicationError);

                HistoryDataValueSeqType returnReport
                        (in unsigned short howMany,
                        out HistoryDataIterator resultIterator)
                        raises (itut_x780::ApplicationError);

        };  // interface AbstractHDScanner




/**
```

5.5.6 ListSelectionHDScanner

```
See section 4.3.6 for behaviour.
*/
        valuetype ListSelectionHDScannerValueType: AbstractHDScannerValueType
        {
                CurrentDataNameSetType          currentDataInstanceList;
                        // GET-REPLACE
        }; // valuetype ListSelectionHDScannerValueType


        interface ListSelectionHDScanner: AbstractHDScannerValueType
        {
/**
This attribute is supported by Get, Set, Add, and Remove access operations.
*/
                CurrentDataNameSetType currentDataInstanceListGet ()
                        raises (itut_x780::ApplicationError);

                void currentDataInstanceListSet
                        (in CurrentDataNameSetType dataInstanceList)
                        raises (itut_x780::ApplicationError);

                void currentDataInstanceListAdd
```

```
                        (in CurrentDataNameSetType dataInstanceList)
                        raises (itut_x780::ApplicationError);

                void currentDataInstanceListRemove
                        (in CurrentDataNameSetType dataInstanceList)
                        raises (itut_x780::ApplicationError);

        }; // interface ListSelectionHDScanner


        interface ListSelectionHDScannerFactory: ManagedObjectFactory
        {
                ListSelectionHDScanner create
                        (in NameBindingType nameBinding,
                        in MONameType superior,
                        inout Istring name, // auto naming if empty string
                        in HDFileFormatType fileFormat,
                        in TimePeriodType fileRetentionDuration,
                        CurrentDataNameSetType currentDataInstanceList)
                        raises (itut_x780::ApplicationError,
                                itut_x780::CreateError);

        };  // interface ListSelectionHDScannerFactory
```

/**

5.5.7 ScopedFilteredHDScanner

See section 4.3.7 for behaviour.
*/

/**

5.6 Notifications

No model specific notifications defined in this document.
*/

/**

5.7 Name Binding

*/

/**
The following module contains name-binding information.
*/

```
        module NameBinding
        {
```

```
/**
```

5.7.1 ThresholdData

```
This name binding is used to name the ThresholdData object to a Managed
Element object.
*/
                module ThresholdData_ManagedElement
                {
                        const string    superiorClass =
                                "itut_m3120::ManagedElement";
                        const boolean   superiorSubclassesAllowed = TRUE;
                        const string    subordinateClass =
                                "itut_q8xx::ThresholdData";
                        const boolean   subordinateSubclassesAllowed = TRUE;
                        const boolean   managerCreatesAllowed = TRUE;
???
                        const DeletePolicyType deletePolicy =
                                itut_x780::deleteOnlyIfNoContainedObjects;
                        const string    kind = "ThresholdData";
                }; // module ThresholdData_ManagedElement
```

```
/**
```

5.7.2 HDFile

```
This name binding is used to name the HDFile object to an AbstractHDScanner
object.
*/
                module HDFile_AbstractHDScanner
                {
                        const string    superiorClass =
                                "itut_q8xx::AbstractHDScanner";
                        const boolean   superiorSubclassesAllowed = TRUE;
                        const string    subordinateClass =
                                "itut_q8xx::HDFile";
                        const boolean   subordinateSubclassesAllowed = TRUE;
                        const boolean   managerCreatesAllowed = FALSE;
                        const DeletePolicyType deletePolicy =
                                itut_x780::deleteOnlyIfNoContainedObjects;
                        const string    kind = "HDFile";
                }; // module HDFile_AbstractHDScanner
```

```
/**
```

5.7.3 AbstractHDScanner

```
This name binding is used to name the AbstractHDScanner object to a
ManagedElement.
*/
                module AbstractHDScanner_ManagedElement
                {
                        const string    superiorClass =
                                "itut_m3120::ManagedElement";
                        const boolean   superiorSubclassesAllowed = TRUE;
```

```
                        const string    subordinateClass =
                                "itut_q8xx::AbstractHDScanner";
                        const boolean   subordinateSubclassesAllowed = TRUE;
                        const boolean   managerCreatesAllowed = TRUE;
                        const DeletePolicyType deletePolicy =
                                itut_x780::deleteOnlyIfNoContainedObjects;
                        const string    kind = "HDScanner";
                }; // module AbstractHDScanner_ManagedElement


        }; // module NameBinding




/**
```

5.8 HDFileFormatConst

```
This module contains constant values identifying history data file formats.
*/
        module HDFileFormatConst
        {
                const string moduleName =
                        "itut_q8xx::HDFileFormatConst";

                const short unknown = 0;

                const short q8xxFileText = 1;

                const short q8xxFileXML = 2;

        }; // HDFileFormatConst


}; // module itut_q8xx



#endif // _itut_q8xx_idl_
```

Annex A (normative): File Format

This annex describes the format of performance measurement data file, named q8xxFileText.

A. 1 Text File Format BNF

The BNF in this annex defines the format of performance measurement data file, named "q8xxFileText".

```
<File> := #file <FD> <FileType> <Newline>
        <Nodes> #endfile <Newline>

<FileType> := <FileFormat> <FD> <FormatVersion> <FD> <InfoModelDomain>

<FileFormat> := <String>

<FormatVersion> := <String>

<InfoModelDomain> := <String>

<Nodes> := <Node> <Nodes>
        | <Node>

<Node> := #node <FD> <NodeID> <FD> <NamePrefix> <Newline>
        <Tables> #endnode <Newline>

<NodeID> := <String>

<NamePrefix> := <Name>
        | <Empty>

<Tables> := <NameAliases> <Table> <Tables>
        | <NameAliases> <Table>

<NameAliases> := <NameAlias> <NameAliases>
        | <NameAlias>
        | <Empty>

<NameAlias> := #namealias <FD> <ShortName> <FD> <LongName> <Newline>

<ShortName> := <Name>

<LongName> := <Name>

<Table> := #table <FD> <Header> <RecordGroups> #endtable <Newline>

<Header> := #header <FD> <MeasuredObjectType> <FD> <GranualarityPeriod>
        <Newline> <DataSetTypes> #endheader <Newline>

<MeasuredObjectType> := <String>

<GranualarityPeriod> := <TimePeriodValue>

<DataSetTypes> := <DataSetType> <DataSetTypes>
        | <DataSetType>

<DataSetType> := #dataset <FD> <DataSetType> <Newline>
        suspect <FD> <ParameterTypes> <Newline> #enddataset <Newline>
```

```
<DataSetType> := <String>

<ParameterTypes> := <ParameterType> <ParameterTypes>
        | <ParameterType>

<ParameterType> := <String> <FieldSeparator>

<RecordGroups> := <RecordGroup> <RecordGroups>
        | <RecordGroup>

<RecordGroup> := #period <FD> <PeriodEndTime> <Newline>
        <Records> #endperiod <NewLine>

<PeriodEndTime> := <TimeValue>

<Records> := <Record> <Records>
        | <Record>

<Record> := <MeasuredObjectName> <FD> <DataSetValues> <Newline>

<MeasuredObjectName> := <Name>  // can be aliased and prefixed

<DataSetValues> := <DataSetValue> <DataSetValues>
        | <DataSetValue>

<DataSetValue> := <Suspect> <FD> <ParameterValues>

<Suspect> := <BooleanValue>

<ParameterValues> := <ParameterValue> <ParameterValues>
        | <ParameterValue>

<ParameterValue> := <Value> <FD>

<Value> := <CounterValue>
        | <GaugeValue>
        | <TidemarkValue>
        | <BooleanValue>
        | <EnumValue>
        | <TimeValue>
        | <TimePeriodValue>

<FD> := :               // field delimiter

<EscapeCharacter> := \

<Newline> := '\n'       // line delimiter

<Name> := <String>

<CounterValue> := <IntegerString>

<GaugeValue> := <FloatString>

<TidemarkValue> := <FloatString>

<BooleanValue> := <BooleanString>

<EnumValue> := <IntegerString>

<TimeValue> := <String> // UTC in format "YYYYMMDDHHMMSS.fffZ"
```

```
<TimePeriodValue> := <IntegerString> <TimePeriodType>

<BooleanString> := T | F        // true or false

<TimePeriodType> := days | hours | minutes | seconds

<String> := {ISO 8859-1(Latin-1) charatcters}

<IntegerString> := <IntegerString> <Digit>
        | <Digit>

<Digit> := 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

<FloatString> := <Mantissa> | <Mantissa> <Exponent>

<Mantissa> := <IntegerString> | <IntegerString> . | . <IntegerString>
         | <IntegerString> . <IntegerString>

<Exponent> := <Exp> <Sign> <IntegerString>

<Sign> := + | -

<Exp> := E | e
```

Notes:

1. White space in the file is meaningless. File generator shall not put white space into the file. File parser shall skip and ignore any white space encountered in the file.
2. A backward compatible file parser shall skip and ignore any data between unknown #<tag> and #end<tag>.
3. Extension to this file format shall require new #<tag> definitions.

A. 2 Text File Format Description

The history data are contained in performance measurement file. The file consists of a file type definition and a set of measurement data. The order of the data is insignificant. There shall be at least one set of data in the file. Otherwise, file shall not be created.

The performance measurement file shall follow these rules for file generating. (See the figure below which depicts the file structure):

1. The file type is defined by its file format, format version, and information modeling domain. FileFormat indicates which file format this file is followed. And FormatVersion specifies a version of particular file format. The value for FileFormat of this recommendation is "q8xxFileText", and FormatVersion is "version1".
2. The information modeling domain is domain of management system which the file is generated from. If the file is generated from a CORBA management system compliant with ITU-T CORBA framework, the value for InfoModelDomain shall be "CORBA". Otherwise, an appropriate value shall be provided, for instance, "CMIP", "SNMP", "TL1", "abcCORBA", etc.
3. The performance management specification for a specific information modeling domain shall specify the semantics and naming convention of

InfoModelDomain, NodeID, Name, NamePrefix, MeasuredObjectName, MeasuredObjectType, DataSetType, ParameterType.

4. NamePrefix defines the common prefix for MeasuredObjectName in a node section.
5. NameAlias defines the short alias name for long FDN name. The scope of an alias definition extends to end of file unless re-defined by another NameAlias.

```
#file:FileFormat:FormatVersion:InfoModelDomain
    #node:NodeID1:NamePrefix  // name prefix is optional
            NameAliases // optional
        #table
                #header:MeasuredObjectType:GranularityPeriod
                        #dataset:DataSetType1
                                suspect:ParameterType1:ParameterType2:…:
                        #enddataset
                        #dataset:DataSetType2
                                suspect:ParameterTypeA:ParameterTypeB:…:
                        #enddataset
                        …
                #endheader
                #period:PeriodEndTime1
                        MeasuredObjectName1:F:123:234:…:T:345:456:…:…:
                        MeasuredObjectName2:F:234:345:…:F:456:567:…:…:
                        …
                #endperiod
                #period:PeriodEndTime2
                        MeasuredObjectName1:F:456:345:…:F:234:123:…:…:
                        MeasuredObjectName2:F:567:456:…:F:345:234:…:…:
                        …
                #endperiod
        #endtable
            NameAliases // optional
        #table
        …
        #endtable
            …
    #endnode
    #node:NodeID2:NamePrefix // name prefix is optional
    …
    #endnode
    …
#endfile
```

Figure 6. PM File Format

Annex B (normative): CORBA File Format Convention

This section describes the semantics and naming convention of InfoModelDomain, NodeID, Name, NamePrefix, MeasuredObjectName, MeasuredObjectType, DataSetType, ParameterType in ITU-T CORBA framework.

1. InfoModelDomain – shall be "CORBA".
2. NodeID – FDN of managed element, or application specific name of managed element.
3. Name – FDN of CORBA name of managed object.
4. NamePrefix – RDN of CORBA name of managed object.
5. MearsuredObjectName – FDN of measured managed object.
6. MeasuredObjectType – Scoped interface name of measured managed object.
7. DataSetType – Scoped interface name of CurrentData or its subclass.
8. ParameterType – Name of attribute in HistoryDataValueType or its sub-valuetype.

Annex C (informative): Example of CurrentDataValueType and HistoryDataValueType inheritance usage

As described in Section 4.2.1, HistoryDataValueType is used as the read-only historical data stores of current data. This valuetype will be accessible through methods on the CurrentData interface. When a current data is subclassed it is intended that the corresponding HistoryDataValueType be subclassed along with it.

This annex provides an example to show how to define a HistoryDataValueType given a CurrentData Interface and how to access history data values through the corresponding CurrentData instance.

In this example; the trafficControlCurrentData interface is used; which is for monitoring the effectiveness of a traffic control. The interface is given below:

```
interface TrafficControlCurrentData: CurrentData
{
/**
the number of calls which are actually affected by the traffic control
instance.
*/
        short callsAffectedByTrafficControlGet ()
                raises (itut_x780::ApplicationError);

/**
the number of calls which were offered to the traffic control instance.
*/
        short callsOfferedToTrafficControlGet ()
                raises (itut_x780::ApplicationError,
                        NOcallsOfferedPackage);
}; // interface TrafficControlCurrentData
```

Based on the CORBA NM Framework, in order to receive all of the attributes supported by an instance of TrafficControlCurrentData, a TrafficControlCurrentDataValueType shall be defined:

```
valuetype TrafficControlCurrentDataValueType: CurrentDataValueType
{
        public short callsAffectedByTrafficControl;
                // trafficControlCurrentDataPackage
                // GET
        public short callsOfferedToTrafficControl;
                // conditional
                // callsOfferedPackage
                // GET
}; // valuetype TrafficControlCurrentDataValueType
```

And to save the measurements collected in a TrafficControlCurrentData instance, a TrafficControlHistoryDataValueType shall be defined:

```
valuetype TrafficControlHistoryDataValueType: HistoryDataValueType
{
        public short callsAffectedByTrafficControl;
                // GET
        public short callsOfferedToTrafficControl;
                // GET
}; // valuetype TrafficControlHistoryDataValueType
```

Given the definition of the TrafficControlCurrentData interface and the TrafficControlHistoryDataValueType, an instance of TrafficControlCurrentData interface shall be created at the runtime to monitor the effectiveness of a particular traffic control.

At the end of each granularity period, a record of TrafficControlHistoryDataValueType is generated and the measurements. callsAffectedByTrafficControl and callsOfferedToTrafficControl collected in the instance shall be copied over to the history record.

To access TrafficControl history data values call one of the two methods: getMostRecent() and getBetween() from the instance of TrafficControlCurrentData.

To access history data values in broader scope use HistoryDataScanner instance.

1. add on demand file (for kick only, i.e. no change of data set) in HDScanner
2. add on demand iterator in HDScanner because of valuetype
3. add history data scope and filter (look at EGPL26 Q.821)
4. fix UML
5. make idl comment short by reference to document section.
6. add Façade version
7. fix the example make it more real (scoped name, DN)
8. Line-oriented text file format (ISO 8859-1 latin-1).
9. Binary-to-ascii-to-binary
10. string, integer, float (E notation), date and time encoding, time interval encoding, enum string, boolean
11. requirement in document
12. justification for removal of Q.822 feature of scan reports delivered in notifications.
13. add GaugeType, CounterType, TideMark, (x.721, X.738, X.739 generic types)
14. design requirement for currentData grouping
15. container object and its related currentData for HDScanner.
16. fileLocation should not re-useable in a short time period.
17. refine applicationError code for file not support.
18. add fileRetentionTime just as historyRetentionTime.
19. add fileRetentionTime into HDFile.
20. look into Q.816.1 iterator deletion statement for file deletion behaviour.
21. add statement for creation of factory in example.

T1M1.5/2001-081R1

STANDARDS PROJECT: Management Services

TITLE: Proposal For A New Kind Of History Data Scanner

SOURCE: AT&T

D. L. Matthews

C1-2C12

200 S. Laurel Ave.

Middletown, NJ 07748

Phone: (732) 420-1613

FAX: (732) 368-1911

Email: dlmatthews@att.com

ISSUES ADDRESSED: TMN, Performance Monitoring, CORBA

DATE: June 8, 2001

DISTRIBUTED TO: T1M1.5

ABSTRACT

This contribution proposes a new kind of history data scanner that would select report data based on Current Data class and grandularity

NOTICE: This contribution has been prepared by AT&T and Telcordia to assist the Standards Committee as a basis for discussion. This should not be construed as a binding proposal on AT&T or Telcordia. Specifically, AT&T and Telcordia reserves the right to request amendments and modifications to the proposal in the future.

UIT - Secteur de la normalisation des télécommunications
ITU - Telecommunication Standardization Sector
UIT - Sector de Normalización de las Telecomunicaciones

Study Period 2001-2004

Commission d'études / Study Group / Comisión de Estudio } **4**

Contributi on tardive / Delayed Contributi on / Contribuci ón tardía } **D.xxx**

Geneva, 2 - 13 July 2001

Texte disponible seulement en / Text available only in / Texto disponible solamente en } **E**

Question(s): 19/4

SOURCE*: T1M1 (United States of America - proposed)

TITLE: Proposal For A New Kind Of History Data Scanner

_____

Abstract

This contribution proposes a new class of history data scanner that selects data based on monitored object class, current data class and grandularity

* **Contact:** David Matthews
AT&T

Tel: +1 732 420 1613
Fax: +1 732 368 1911
E-mail: dlmatthews@att.com

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

Introduction

At the last Joint CORBA EG meeting a new requirement was discussed that is difficult for the current Q.822 CORBA model to met. The new requirement is:

The model shall support a periodic performance report that will incude performance data on new resources added to the system automaticly. That is to say, the report will include data on new resources added to the system without the manager needing to re-administer the report definition.

This requirement allows the manger to receive comprehensive performance information on all relivant resources in envirorments where there is frequent provisioning activity with the manager constantly administering the performance report definition.

In the model periodic reports are administered by history data scanner interfaces. There are currretly two instanable classes defined, ListSelectionHistdataScanner and ScopedFilteredHistoryDataScanner.

The ListSelectionHistdataScanner selects information for the report with a fixed list of current data object instances. This kind of scanner does not support this new requirment. It's use would require the manager to administer the list to add the current data object instance names to the list for any new resources added to the system.

The ScopedFilteredHistoryDataScanner selects information for the report by selecting current data managed objects using a scope and filter. The behavior of this object needs to be clairified. It is not clear when the scope and filter is to be run. There seem to be two possibilities. One is for the scope and filter to be run when requested by the manager, perhaps by having the parameters assoitated with the scope and filter changed by the manager. This type of behaviour would not support the proposed new requirement. A more reasonable understanding of the behavious would be for the scope and fileter to run once each granularity period. This would meet the new requirement since new resouces could be captured the the scope and filter. However, running any significant number of scopes and filters frequently, could provide to be computationaly bundensome. In addition, an optimized implimentation of this kind of selection could be difficult. This kind of history data scanner should remain in the model but a simpler alternative that could meet the new requirement should also be provided.

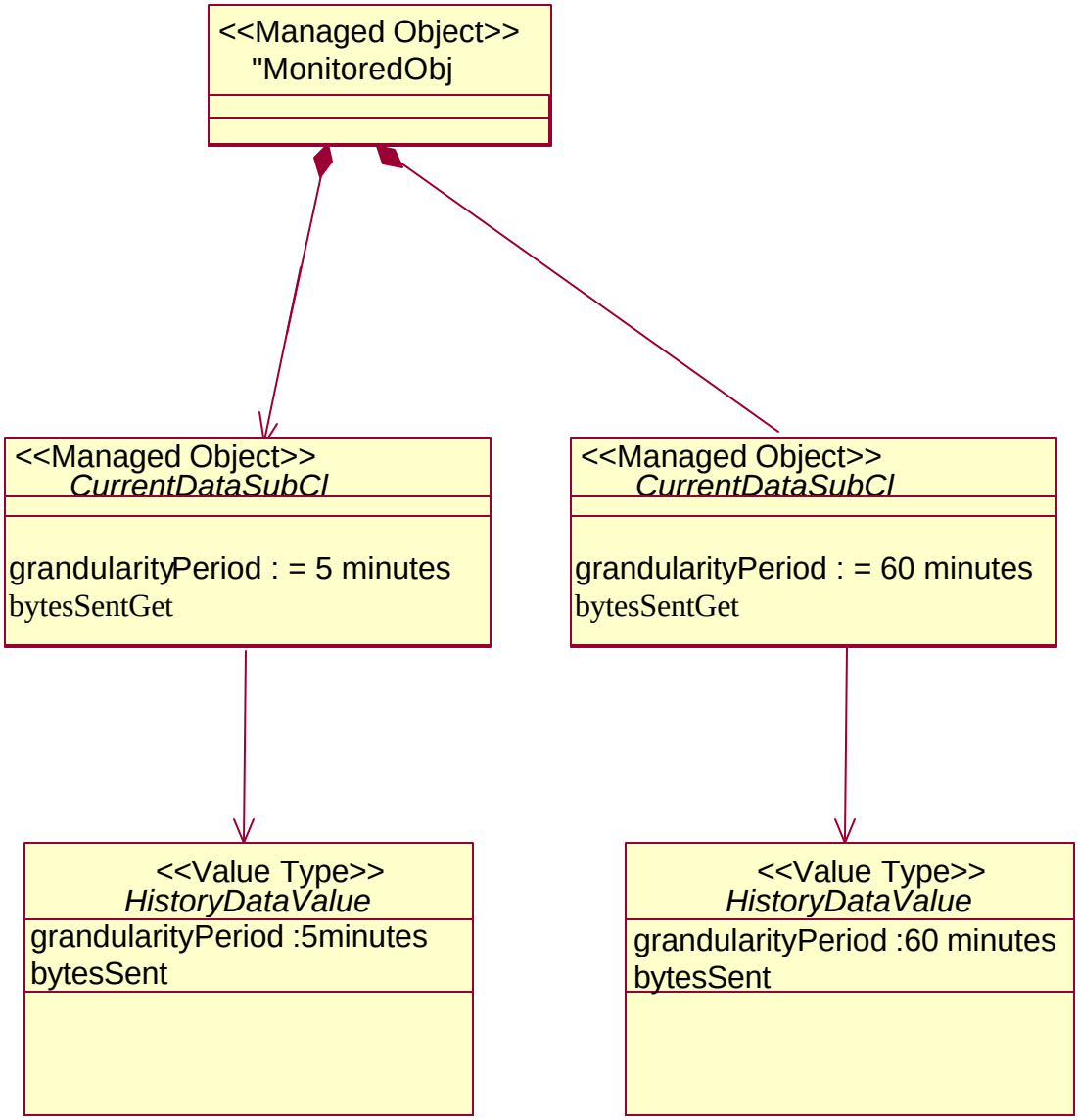
This paper proposes a simple kind of history data scanner that could meet the requirement. This would be a history data scanner that selected data to be scanned by indicating the class of current data managed object that have their related history data included in the report. In addition to the class, the grandularity of the current data would also have to be indicated since (as shown in Figure 1) multiple instances of the same class of current data can be contained under a single monitoried object with different grandularites and the manager must be able to choice which grandularity to include in the performance report.

Also, since in many application, the same class of current data is used to measure the performance of different classes of measured objects (for example the traffic current data class in ATM) a list of measured object classes is also included in this new history data scanner.

It is purpose that this new history data scanner have three set valued attributes

- One giving an administerable list of measured object classes,
- One giving an administerable list of CD subclasses,
- An another giving a list of grandularities.

Data would be included in the report that is associated with one of the listed measured classes and one of the current data classes having one of the listed gradularites. An empty list is to be interpated as indicating all. That is a empty MO list indicates all MO and therefore all listed CD classes are selected. Simialarly, an empty CD class indicates all CDs (with one of the listed gradularities). Note: if all lists are empty, all current data objects in the system are selected. A UML class diagram of the new class is shown in Figure 2.


<<Managed Object>>
"MonitoredObj
<<Managed Object>>
CurrentDataSubCl
grandularityPeriod : = 5 minutes
bytesSentGet
<<Managed Object>>
CurrentDataSubCl
grandularityPeriod : = 60 minutes
bytesSentGet
<<Value Type>>
HistoryDataValue
grandularityPeriod :5minutes
bytesSent
<<Value Type>>
HistoryDataValue
grandularityPeriod :60 minutes
bytesSent

Figurre 1

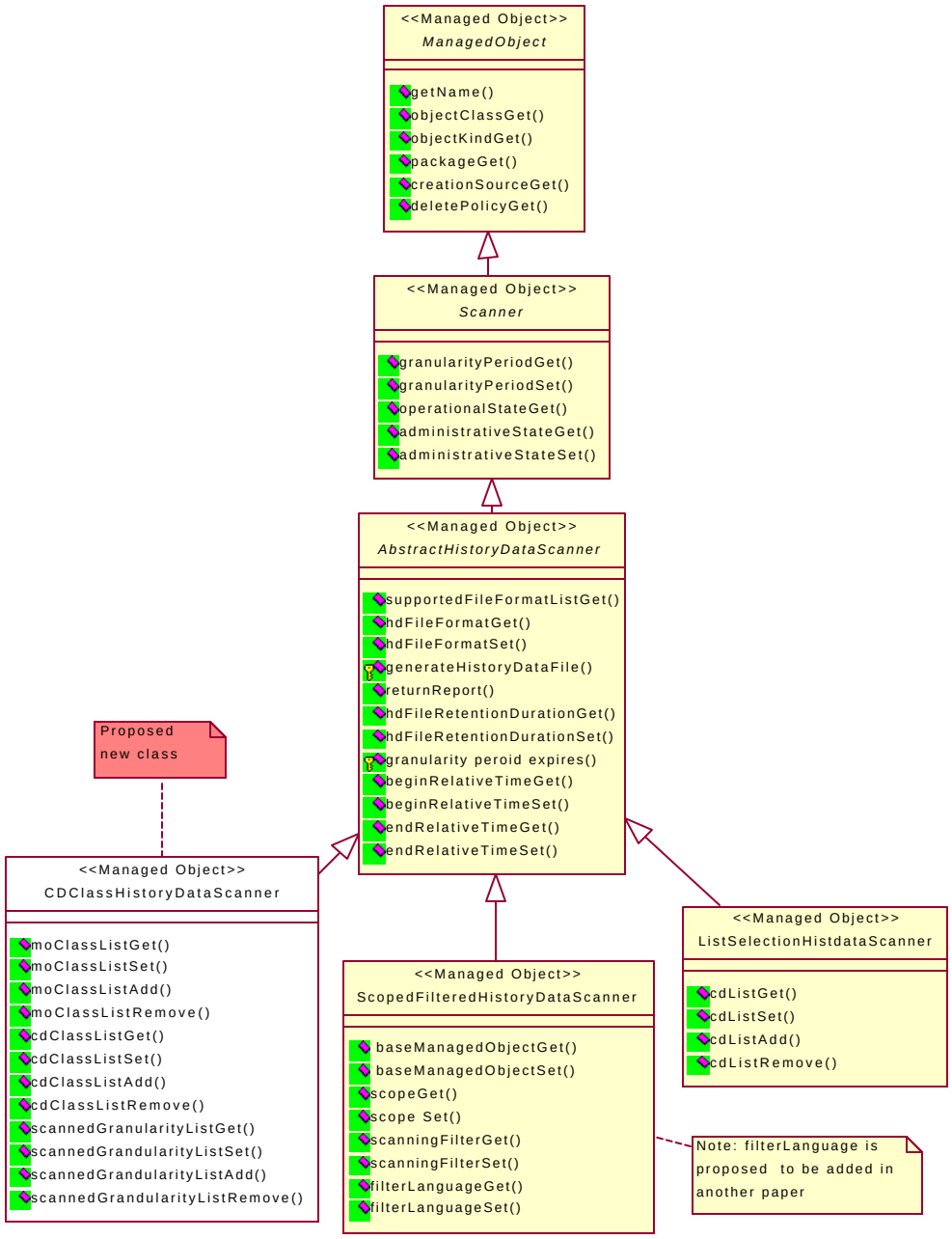

<<Managed Object>>
ManagedObject
getName()
objectClassGet()
objectKindGet()
packageGet()
creationSourceGet()
deletePolicyGet()
<<Managed Object>>
Scanner
granularityPeriodGet()
granularityPeriodSet()
operationalStateGet()
administrativeStateGet()
administrativeStateSet()
<<Managed Object>>
AbstractHistoryDataScanner
supportedFileFormatListGet()
hdFileFormatGet()
hdFileFormatSet()
generateHistoryDataFile()
returnReport()
hdFileRetentionDurationGet()
hdFileRetentionDurationSet()
granularity peroid expires()
beginRelativeTimeGet()
beginRelativeTimeSet()
endRelativeTimeGet()
endRelativeTimeSet()
Proposed
new class
<<Managed Object>>
CDClassHistoryDataScanner
moClassListGet()
moClassListSet()
moClassListAdd()
moClassListRemove()
cdClassListGet()
cdClassListSet()
cdClassListAdd()
cdClassListRemove()
scannedGranularityListGet()
scannedGrandularityListSet()
scannedGrandularityListAdd()
scannedGrandularityListRemove()
<<Managed Object>>
ScopedFilteredHistoryDataScanner
baseManagedObjectGet()
baseManagedObjectSet()
scopeGet()
scope Set()
scanningFilterGet()
scanningFilterSet()
filterLanguageGet()
filterLanguageSet()
<<Managed Object>>
ListSelectionHistdataScanner
cdListGet()
cdListSet()
cdListAdd()
cdListRemove()
Note: filterLanguage is
proposed to be added in
another paper

Figure 2.

UIT - Secteur de la normalisation des télécommunications
ITU - Telecommunication Standardization Sector
UIT - Sector de Normalización de las Telecomunicaciones

Study Period 2001-2004

Commission d'études / Study Group / Comisión de Estudio } **4**

Contributi on tardive / Delayed Contributi on / Contribuci ón tardía } **D.xxx**

Geneva, 2 - 13 July 2001

Texte disponible seulement en / Text available only in / Texto disponible solamente en } **E**

Question(s): 17/4

SOURCE*: T1M1 (United States of America - proposed)

TITLE: Proposed Standard Recommendation using GDMO/ANS.1 mappings to XML schemas and Applied to Configuration Audit Support Function (X.792).

Abstract

This technical contribution provides guidelines for defining tML schemas that describe the syntactical structure of information exchanged between agents and managers in a TMN. This information described using tML schemas could be carried using any choice of technology like CMIP, CORBA (and ValueTypes as payload), HTTP, HTTPS, Interactive Agent (IA), etc. It covers rules for translating models from GDMO/ASN.1, and tML style conventions.

This work is proposed to help the telecommunications Markup Language (tML) Ad Hoc Group (AHG) to identify mechanisms to map legacy GDMO/ASN.1 specifications to the newly proposed tML schemas based information exchange mechanisms.

The research and analysis provided in this document show the feasibility of mapping GDMO/ASN.1 specifications to tML (XML) Schema's following the XML Schema Specifications (W3C Rec., 2 May 2001). This contribution provides an informative document and useful XML Schema's to help the development and analysis of intercarrier eCommerce Applications that make use of the codesets in an XML Schema format.

* **Contact:** Raymond E. Reeves
Sprint

Tel: +1 913 534 3405
Fax: +1 913 534 5136
E-mail: Raymond.E.Reeves@mail.sprint.com

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

CONTENTS

Page

LIST OF FIGURES IV

LIST OF TABLES IV

1 SCOPE 5

1.1 PURPOSE 5
1.2 APPLICATION 6
1.3 ISSUES 7

2 REFERENCES 8

2.1 NORMATIVE REFERENCES 8

3 DEFINITIONS AND ABBREVIATIONS 9

3.1 DEFINITIONS FROM W3C RECOMMENDATION XML SCHEMA PART 1 9
3.2 DEFINITIONS FROM X.680 9
3.3 DEFINITIONS FROM X.701 9
3.4 DEFINITIONS FROM X.792 9
3.5 ABBREVIATIONS 9

4 GDMO/ANS.1 TRANSLATION 10

4.1 MANAGED OBJECT CLASSES 10
4.2 PACKAGES 11
4.3 ATTRIBUTES 12
4.4 ATTRIBUTE GROUPS 12
4.5 ACTIONS 12
4.6 NOTIFICATIONS 12
4.7 BEHAVIORS 12
4.8 NAME BINDINGS 12
4.9 PARAMETERS 13
4.10 ASN.1 MODULES AND DATA TYPES 14
4.10.1 ASN.1 Modules 14
4.10.2 Basic Types 15
4.10.3 Sequence 16
4.10.4 Sequence of 16
4.10.5 Set 16
4.10.6 Set of 17
4.10.7 Choice 17
4.10.8 Object Identifier (OID) 17
4.10.9 Object Instance 17
4.10.10 ANY 17
4.10.11 Tagged Types 18
4.10.12 Bit String 19

5 STYLE IDIOMS FOR TML SPECIFICATIONS 22
5.1 USE CONSISTENT INDENTATION 22
5.2 USE CONSISTENT CASE FOR IDENTIFIERS 22
5.3 DECOUPLE TYPES FROM ELEMENTS 22
5.4 USE A CONSISTENT TYPE SUFFIX 23
5.5 USE A CONSISTENT SUFFIX FOR SEQUENCE TYPES. 23
5.6 USE A CONSISTENT SUFFIX FOR SET TYPES. 23
5.7 USE A CONSISTENT SUFFIX FOR OPTIONAL TYPES 23
5.8 ASSUME NO GLOBAL IDENTIFIER SPACES 23
5.9 GLOBAL LEVEL DEFINITIONS 23
5.10 EXPLICIT VS. IMPLICIT NAMESPACES 23
6 APPLICATION TO THE CONFIGURATION AUDIT SUPPORT FUNCTION 23
6.1 X.792 ASN.1 DEFINITIONS AND CORRESPONDING TML SCHEMAS 24
6.1.1 AuditASN1FileModule 24
6.1.2 AuditASN1Module 29
ANNEX A THE BITSTRING, ANY AND NULL TYPE DEFINITIONS 34
ANNEX B EXAMPLES AND HINTS 35
B.1 EXAMPLE OF A PERSONNEL RECORD 35
B.1.1 Example of a personnel record 35
B.1.2 ASN.1 description and tML schema mapping of the record structure 35
B.1.3 ASN.1 description of record value and corresponding tML instance document 37
B.2 EXAMPLES FOR DATA TYPES IN GENERAL 38

List of Figures

Figure 1. tML Based Specification 6

List of Tables

Table 1. Example of simple ASN.1 BIT STRING Mapping 19
Table 2. Example Mappings to Bit String Sequence Type 20

Re-using a generic information model for a variety of network technologies and network management applications through a mapping to tML schemas will speed the introduction of network services while keeping network management system development costs down.

The telecommunications industry has invested a great deal of time and energy in the development of information models for the CMIP network management protocol. A primary goal of the tML framework is the re-use of these information models by enabling their translation to tML Schemas with little change in semantics. As a result, initial tML schemas are derived from GDMO/ASN.1 models.

1.2 Application

Recommendation M.3020 defines three phases in the development of a TMN specification. The three phases are Requirements, Analysis and Design. Figure 1 shows this process and the scope of this Recommendation for developing tML Schemas based specification relative to this process.

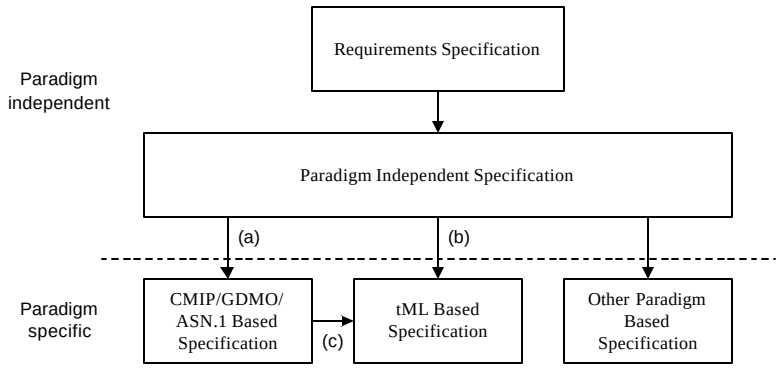


Figure 1. tML Based Specification

The requirements and analysis are specified using an approach that is not specific to a network management technology paradigm. The output from the analysis phase, the paradigm independent specification, is used as input to the paradigm specific design phase.

In the design phase, network management paradigm specific features are used to define information models. These paradigm specific specifications incorporate both behavior (normally in natural language) and formal interface signatures (e.g., GDMO/ASN.1, IDL, tML schemas).

The arrows marked as (a) and (b) show that the analysis output is mapped to a GDMO/ASN.1 based model to use with CMIP or tML Schema models to use with the choice of message transport mechanism, respectively (or IDL models to use with CORBA/IIOP). There are no prescriptive rules available at this time to generate these models. It may be possible to develop such rules in the future in M.3020.

This contribution addresses the reuse of existing models developed in the GDMO/ASN.1/CMIP paradigm if tML schemas are to be used instead of GDMO/ASN.1. The arrow shown as (c) is addressed by this Recommendation.

In developing the transformation from GDMO/ASN.1 definitions to tML schemas two approaches are possible.

- In the first approach, every element of the syntax is translated to tML Schemas using a well-specified algorithm or a prescriptive definition. This approach is the one where a gateway can be used to support interoperability.

- The second approach (used in this contribution) does not prescriptively translate every element of the syntax. Rather, the elements are translated from the existing GDMO/ASN.1 in a way that preserves the semantics. This approach is not used for inter-working via gateways but to

preserve the requirements and semantics of the models developed to meet the telecommunication context. It is applied when the managing and managed systems are designed to communicate using information exchange (instance documents) based on tML Schemas.

This contribution takes advantage of GDMO/ASN.1/CMIP information models. Additional advantages will be described in the tML standard framework. The tML standard framework does not require a GDMO/ASN.1 model to be developed prior to the development of a tML schema. In fact, developing a new tML Schema information model for use within this framework is straightforward and guidelines for doing so are provided.

1.3 Issues

This recommendation, in its current revision, still leaves some open issues that are needed to address in a future revision or as part of another effort. These issues include:

- This recommendation does not provide for request/response/notification semantics and mechanisms much like other tML (XML) transfer protocols (e.g. Simple Object Access Protocol –SOAP–): This contribution only describes the structured information exchanged by a manager and an agent. This information is then used in operations as payload or simply transferred by transport agents (e.g. CORBA objects, Interactive Agents –IA–, etc.).

  As expressed in other tML contributions and discussions "[t]he issue of interoperability of available protocols that can handle [t]ML transport was discussed in the ITU-T SG4 experts meeting on May 21-24, 2001 in Richardson Texas. It was decided that a roadmap is needed to guide the selection of the appropriate protocols used in tML. […]"

- It is open for discussion if the CMIP/CMIS Recommendations should be mapped to tML so that the same remote process communication mechanisms (CMIS primitives and CMIP operations) could be used to convey operations between managers and agents using tML tags instead of ASN.1 encoding (BER, PER, etc.)

- Because this proposed recommendation only addresses the mapping of GDMO/ASN.1 to tML Schemas, the question still remains on how to address other definitions made directly to CORBA IDL. A desired approach is that recommended by the TMN interface specification methodology. In that case, the MIB file structure would be defined using UML notation.

- This proposed recommendation addresses two different problems: mappings to tML and application to X.792 including mapping of several ASN.1 modules from different standards (e.g. M.3100, X.501, X.711, etc.). It is desired to explore the possibility to have each separate aspect addressed in this recommendation into several proposed recommendation. It is, one proposed recommendation for each translation and one recommendation for the mapping guidelines.

- It is necessary to investigate the option of re-factoring most tML schemas and determine how common tags could be exported to the tML standard framework.

- The final conventions used in the tML standards framework will prevail in this proposed recommendation. For example, namespace names and common tags will be reused from the tML standard framework in order to reduce "tag and type space" littering.

2 References

2.1 Normative References

At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all 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.

- ITU-T Recommendation M.3020 (2000), *TMN Interface Specification Methodology*.
- ITU-T Recommendation M.3100 (1995), *Generic Network Information Model*.
- ITU-T Recommendation X.501 (1997) | ISO/IEC 9594-2: 1998 (E), *Information Technology – Open Systems Interconnection – The Directory: Models*.
- ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1: 1998 (E), *Information Technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.
- CCITT Recommendation X.711 (1997 E) | ISO/IEC 9596-1: 1998 (E), *Information Technology – Open Systems Interconnections – Common Management Information Protocol: Specification*.
- CCITT Recommendation X.722 (1992) | ISO/IEC 10165-4: 1992, *Information Technology – Open Systems Interconnections – Structure of Management Information: Guidelines for the Definitions of Managed Objects.*
- ITU-T Draft Recommendation X.780, *Guidelines for the Definitions of CORBA Managed Objects.*
- ITU-T Recommendation X.792 (1999), *OSI management – Management functions and ODMA functions – Configuration audit support function for ITU-T applications*.
- Extensible Markup Language (XML) 1.0, Second Edition, Tim Bray et al., eds,. W3C, 6 October 2000. See http://www.w3.org/TR/2000/REC-xml-20001006.
- Namespaces in XML, Tim Bray et al., eds,. W3C, 14 January 1999. See http://www.w3.org/TR/1999/REC-xml-names-19990114.
- XML Schema Part 1: Structures. Henry Thompson et al., eds,. W3C Recommendation, 2 May 2001. See http://www.w3.org/TR/2001/REC-xmlschema-1-200010502.
- XML Schema Part 2: Datatypes . Paul Biron et al., eds,. W3C Recommendation, 2 May 2001. See http://www.w3.org/TR/2001/REC-xmlschema-2-200010502.

3 Definitions and Abbreviations

3.1 Definitions from W3C Recommendation XML Schema Part 1

The following terms used in this Recommendation are defined in the XML Schema Part 1: Structure (W3C Rec. XML Schema Part 1):

– XML Schema
– Schema component
– Target namespace
– Declaration
– Definition
– Type definition
– Simple type
– Complex type
– Type restriction
– Type extension
– Base type

3.2 Definitions from X.680

The following terms used in this Recommendation are defined in the Abstract Syntax Notation One: Specification of basic notation (ITU-T Rec. X.680):

– ASN.1 specification
– (ASN.1) module
– external reference
– null type
– bitstring type

3.3 Definitions from X.701

The following terms used in this Recommendation are defined in the Systems Management Overview (ITU-T Rec. X.701):

– managed object class
– manager
– agent

3.4 Definitions from X.792

The following terms used in this Recommendation are defined in the Configuration Audit Support Function (ITU-T Rec. X.792):

– MIB file

3.5 Abbreviations

This Recommendation uses the following abbreviations:

ASN.1	Abstract Syntax Notation #1
CMIP	Common Management Information Protocol
CORBA	Common Object Request Broker Architecture

GDMO	Guidelines for the Definition of Managed Objects
GIOP	General Interoperability Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
HTTPS	HTTP Secure
IDL	Interface Definition Language
ITU-T	International Telecommunication Union – Telecom
MIB	Manage Information Base
OAM&P	Operations, Administration, Maintenance, and Provisioning
tML	XML for TMN
TMN	Telecommunications Management Network
UML	Unified Modeling Language
UTRAD	Unified TMN Requirements, Analysis and Design
W3C	World Wide Web Consortium
XML	Extensible Markup Language

4 GDMO/ANS.1 Translation

This section provides guidelines for creating tML schema information models from existing information models described using GDMO/ASN.1. The sections below describe how each of the GDMO templates and ASN.1 types are to be translated to tML schemas.

4.1 Managed Object Classes

If desired, each Managed Object Class in a GDMO specification shall be translated into a tML type containing only the attributes contained in the GDMO template. This is because tML does not carry functional operations but only describes information structure[1]. Translations of Managed Object Classes derived from the GDMO Top class could extend from a *ManagedObject* tML base type (not described in this recommendation).

Translations of classes not derived directly from Top shall inherit from the translation of whatever class they are derived from. All managed object types must inherit directly or indirectly from the *ManagedObject* tML base type. Multiple inheritances or multiple type extension is not allowed in tML Schemas.

The inability to inherit or type extension from a potential superclass/base-type also means manual work may be required if the potential superclass/base-type or any of its super classes/base-types is modified. A more serious issue is that tML schemas type substitution is based on type extension mechanisms and substitution groups. If the type does not inherit/extends from the base-type or the element declaration is not part of the same substitution group as the base-element (head of the substitution group), it cannot be substituted to it. Unfortunately this is a limitation of XML Schemas, not these guidelines.

Operations, actions, and notifications in mandatory and conditional packages are translated into annotations on the tML schema type declaration according to the guidelines below. These are to be defined either in the documentation or the app-info elements within annotation.

[1] In order to allow for operations to be used in an all tML (XML) environment, request/response mechanisms may be defined much like in the Simple Object Access Protocol (SOAP). However, in its current state, this contribution only describes the structured information exchanged by a manager and an agent. This information is then used in operations or simply transferred by transport agents (e.g. CORBA objects, Interactive Agents –IA–, etc.).

However, the base tML base type would contain only operations, which cannot be mapped to any tML (XML) schema structure[2]. As per ITU-T X.780, the *ManagedObject* interface contains seven operations (i.e., nameGet, objectClassGet, packagesGet, creationSourceGet, deletePolicyGet, attributesGet, destroy). This recommendation can be used to define, in terms of tML (XML) schemas, the structure of the return values used in each of these seven operations.

For instance, the *ManagedObjectValueType* return value for the *attributesGet*() operation on any ITU-T X.780 derived managed object could consist of a tML instance document containing all the attributes requested. This tML instance document is sent across as a CORBA *valuetype*. Thus, when a X.780 CORBA-based client invokes the *attributesGet*() operation on any X.780 CORBA-based managed object, it will receive back a reference to a *ManagedObjectValueType* (containing a tML instance document) which it may then parse to extract the values corresponding to the attributes requested.

Because GDMO allows for conditional packages and attributes to be passed and because in tML schemas only the elements required are transferred provided the schema allows it, the use of tML instance documents as return values is advantageous (as compared to the IDL value types approach used in X.780 where the value type structure is pre-defined and filled-in values are used).

4.2 Packages

Unfortunately, tML schemas do not provide a means of defining packages in one place other than by translating a package into an element group and its operations in an annotation. This results in a large number of element groups and annotations and increases the complexity of the tML schema.
As described above, whenever a GDMO package is included in a Managed Object Class, the translation of that class to a tML type declaration includes a translation of each of the templates in the package.

GDMO attributes that are part of a conditional package shall be translated into an element group contained in a complex element named as the package. GDMO actions and notifications shall be translated into annotations for the new complex element named as the package.

The *present if* clause in the GDMO object's conditional package statement shall be translated to a minOccurs="0" attribute in the new complex element named after the translated package.

Capability collision is not an issue when translating from CMIP because tML schemas do not carry operations and all elements will be contained within package-named elements.

If a GDMO template occurs in multiple conditional packages included in a single object, the modeler may want to consider making the capability mandatory or defining a new conditional packages for the capability. The new package is then mapped to tML as mentioned above.

Note that using exceptions to represent packages only supports conditional packages. If multiple mandatory packages are present in a GDMO class, their description will be separated within the tML type definition as element contained in the content-model defined by the type declaration.

[2] Extending this document to allow request/response semantics is left for further study.

Behavior statements accompanying a package definition shall be translated to comments in the tML schema definitions of the tML types translated from the GDMO objects that include the package.

Registration of packages is not required. They are unique in the namespace in which they are defined.

4.3 Attributes

As described above, GDMO managed object classes list the packages that are to be included in the class definition. The package then lists the attributes, actions, and notifications that make up that package. When translating a managed object class, the included packages will be translated to elements within the managed object (complex) type, and most of these will include attribute definitions.

Unfortunately, tML (as initially proposed) cannot carry the GET, REPLACE, ADD, REMOVE semantics found in GDMO. Thus, the type for the attributes defined in the packages will be a translation of the attribute's ASN.1 syntax.

Default values are defined through default attributes in the element declarations within a tML simple type declaration. It will often make sense to define a tML type for each attribute. Even if the attribute is a simple type, tML <simpleType> or <complexType> statement may be used to define a type for it. An annotation within the type definition for an attribute is the best place to put a translation of an attribute's behavior statement.

The standard tML element and type tags defined by these guidelines shall be used whenever possible.

Registration of attributes is not required. They are unique in the namespace in which they are defined.

4.4 Attribute Groups

GDMO attribute groups will be translated into named groups in tML schemas.

4.5 Actions

Actions shall be translated as annotations in the tML schema.

4.6 Notifications

The 15 notifications found in ITU-T Rec. X.721, which are the notifications used in most GDMO information models are documented in annotations within the corresponding type definition.

If a new notification must be defined it should be added to the type declaration's annotation element.

4.7 Behaviors

GDMO behavior templates shall be translated to annotations within the type declaration.

4.8 Name Bindings

Containment is a very important relationship in network management. In the TMN tML-based framework, this could be represented in a series of ways. Because this recommendation only suggests the definition of the information structure, use of annotation/documentation is the recommended approach.

However, in order to align this recommendation with the approach used in ITU-T X.780, an alternative approach involves the use of a nameBindings element contained in the namespace of the tML schema and with all the attributes and elements fixed. See for example the following tML schema that is a translation of the CORBA TMN IDL module itut_m3120 found in ITU-T X.780 §6.8:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"

    id="tML-ITU-T.M.3120.xsd"
    version="1.0"

    targetNamespace="http://www.itu.int/tML/tML-ITU-T.M.3120"
    xmlns:tML-ITU-T.M.3120="http://www.itu.int/tML/tML-ITU-T.M.3120"
    xmlns:tML-Base="http://www.itu.int/tML/tML-Base"
>
    <!-- etc -->
    <element name="NameBindings">
        <complexType>
            <sequence>
                <element name="Equipment_ManagementElement">
                    <complexType>
                        <sequence>
                            <element name="superiorClass"                    type="string" fixed="itut_m3120::ManagedElement"/>
                            <element name="superiorSubclassesAllowed"        type="string" fixed="TRUE"/>
                            <element name="subordinateClass"                 type="string" fixed="itut_m3120::Equipment"/>
                            <element name="subordinateSubclassesAllowed"     type="string" fixed="TRUE"/>
                            <element name="managerMayCreate"                 type="string" fixed="TRUE"/>
                            <element name="deletePolicy"                     type="string"
                                                                             fixed="DeleteOnlyIfNoCOntainedObjects"/>
                            <element name="kind"                             type="string" fixed="Equipment"/>
                        </sequence>
                    </complexType>
                </element>
                <element name="Equipment_Equipment">
                    <complexType>
                        <sequence>
                            <element name="superiorClass"                    type="string" fixed="itut_m3120::Equipment"/>
                            <element name="superiorSubclassesAllowed"        type="string" fixed="TRUE"/>
                            <element name="subordinateClass"                 type="string" fixed="itut_m3120::Equipment"/>
                            <element name="subordinateSubclassesAllowed"     type="string" fixed="TRUE"/>
                            <element name="managerMayCreate"                 type="string" fixed="TRUE"/>
                            <element name="deletePolicy"                     type="string"
                                                                             fixed="DeleteOnlyIfNoCOntainedObjects"/>
                            <element name="kind"                             type="string" fixed="Equipment"/>
                        </sequence>
                    </complexType>
                </element>
            </sequence>
        </complexType>
    </element>
    <!-- etc -->

</schema>
```

4.9 Parameters

GDMO parameters provide extensibility for GDMO information models. Parameter templates are used to augment an existing specification in the areas of notifications, actions (requests, responses, and failures), and specific errors when defining subclasses. The GDMO definitions of all notifications and

many actions contain an extensibility field that is further defined by the subclasses (if required). In the case of specific errors, class-specific errors are used to augment the general "processing failure" error in CMIP. The format of this information is often a list of name-value pairs, where the name defines the data type of the value.

There are several keywords used in GDMO parameter templates to specify the semantics of the extensions: ACTION-INFO, ACTION-REPLY, EVENT-INFO, EVENT-REPLY, Context-Keyword, SPECIFIC-ERROR.

GDMO parameters can only be translated to tML schema annotations.

4.10 ASN.1 Modules and Data Types

GDMO uses the ASN.1 language to define the syntax of attributes as well as operation and notification parameters. These syntax specifications are contained in an ASN.1 specification which, in its turn, contains one or more ASN.1 modules. When converting GDMO templates to tML Schema, these syntax definitions must be translated. This section gives guidelines on translating ASN.1 syntax to tML Schema elements and types.

4.10.1 ASN.1 Modules

ASN.1 Modules are used for organizing a collection of instances of the use of the ASN.1 notation for type, value, etc., encapsulated using the ASN.1 module notation. In effect, an ASN.1 module defines a name space for all the ASN.1 definitions contained in it. ASN.1 module names are used in external references for referring to definitions contained in other modules from the referring module. In essence, the ASN.1 module concept maps to the tML Schema and namespace.

This recommendation proposes mapping ASN.1 modules (considering the identifiers) to tML Schemas with a target namespace so that namespace prefixes could be used in external references. The DefinitiveIdentifier uniqueness semantics are resolved by the uniqueness provided by the tML namespaces URI. The ASN.1 IMPORT semantics shall be mapped to the "import" mechanism provided by tML Schemas. Because tML Schema does not allow multiple default namespaces, it is required that every imported definitions be prefixed with the corresponding namespace prefix. In order to have type definitions and element descriptions to be "EXPORTed" or visible form another schema, it is required to have them at the global level within the tML schema.

It must also be known that ASN.1 mapped tML schemas shall import the tML schema that defines the non conventional types used to allow the ASN.1 to tML mapping. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    ...
    ...
    xmlns:tML-X.680="http://www.itu.int/tML/tML-X.680"
>
    ...
    ...
    <import namespace="http://www.itu.int/tML/tML-X.680" schemaLocation="tML-X.680.xsd"/>
    ...
    ...
</schema >
```

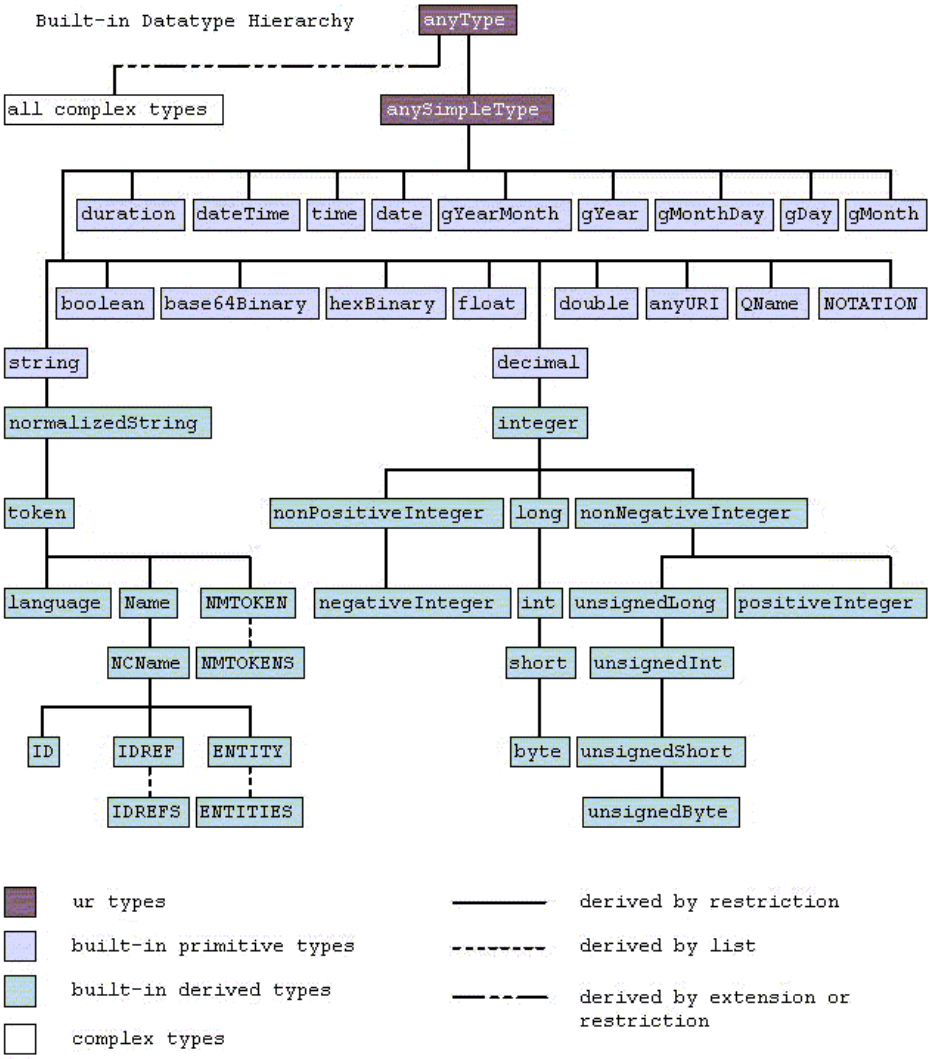
The ASN.1 TAG nor module registration semantics are not needed in tML Schema.
For an example on how to apply the above recommended mapping see Section 6.

4.10.2 Basic Types

tML Schema defines the following hierarchy of built-in basic datatypes (a.k.a 'types'), derived from and defined in the XML Schemas specification[3], to which ASN.1 basic types may be translated:

As per the W3C XML Schema Specification, "[p]rimitive datatypes are those that are not defined in terms of other datatypes; they exist ab initio." And, "[d]erived datatypes are those that are defined in terms of other datatypes."

Some of the built-in datatypes include: anyType, boolean, byte, double (for double-precision floating-point numbers), float (for single-precision floating-point numbers), long (for large integers), object (for object references), integer, short (for small integers), hexBinary, string, and Char (as defined in XML 1.0 2nd edition). Enumerations, which are a basic type in many languages (including IDL) is obtained through the enumeration constraining facet that could be applied to most built-in types.



This guideline uses the *string* type for all strings. The actual character encoding is defined in the encoding declaration contained in the xml processing instruction. In an encoding declaration, the values "UTF-8", "UTF-16", "ISO-10646-1", "ISO-8859-2", …, "ISO-8859-9" should be used for the various encodings and transformations of Unicode / ISO/IEC 10646, the values "ISO-8859-1", "ISO-8859-2", …, "ISO-8859-9" should be used for the parts of ISO 8859, and the values "ISO-2022-JP", "Shift_JIS", and "EUC-JP" should be used for the various encoded forms of JIS X-0208-1997.

[3] See XML Schema Part 2: Datatypes, W3C Recommendation 2 May 2001, §3.

A sample declaration of wide character encoding follows:

```
<?xml version="1.0" encoding="UTF-16"?>
<            ="                  ">
    <  >      </  >
    <   >                  </    >
    <!--         - ->
    <    >&      ;</    >
    <             ="      "/>
</    >
```

Because of some tML schema limitations, not all the semantics within ASN.1 are easily expressed in tML schemas. In order to simplify and reduce complexity of the mapping guidelines, named versions of most type declarations and values shall not be used, except for bitstring's. The following ASN.1 expressions will not easily map the complete semantics to tML schemas:

a INTEGER ::= 1
T1 ::= INTEGER { a(2) }
T2 ::= INTEGER { a(3), b(a) }
c T2 ::= b
d T2 ::= a

In this case, "a" has three contexts:

1. "a" as an integer with value equal to 1
2. "a" as a named value within T1 context and a value equal to 2
3. "a" as a named value within T2 context and a value equal to 3

For this reason, "d" denotes the value 3, "c" denotes the value 1

This level of complexity, parsing and substitution shall be discouraged in the tML schema declarations.

4.10.3 Sequence

As per ITU-T Rec. X.680 (1997 E), sequence types are types defined by referencing an *ordered* list of types (some of which may be declared to be optional); each value of the sequence type is an *ordered* list of values one, from each component type.

tML schemas support the definition of *ordered* data structures using *sequence* groups defined in complexTypes, similar to ASN.1 *sequence* types.

4.10.4 Sequence of

tML schema supports the definition of "sequence of" types, both basic and complex, in much the same way as the ASN.1 *sequence* type. All needed is to modify the maxOccurs facet to unbounded. It is recommended that the sequence elements be named after their type without the "Type" or " postfixes. For example:

```
<element name="firstTwo" type="x680:NamesOfMemberNationsType"/>
<complexType name="NamesOfMemberNationsType">
    <sequence maxOccurs="unbounded">
        <element name="string" type="string"/>
    </sequence>
</complexType>
```

4.10.5 Set

As per ITU-T Rec. X.680 (1997 E), set types are types defined by referencing a *fixed, unordered* list of distinct types (some of which may be declared to be optional); each value of the set type is an *unordered* list of values, one from each component type.

tML schemas support the definition of *unordered* data structures using *all* groups defined in complexTypes, comparable to ASN.1 *set* types.

However, because children belonging to an “all group” must be individual elements –that is, this group must itself contain no groups–, the set types semantics are not fully represented with *all* groups. Instead, sets are translated to tML schema *sequence* groups. The convention for ending the type name with “SetType” shall be followed. When handling set values, duplicates should be eliminated and order ignored.

4.10.6 Set of

tML schema supports the definition of “set of” types, both basic and complex, in much the same way as the ASN.1 *set* type. All needed is to modify the maxOccurs facet to unbounded. It is recommended that the set elements be named after their type without the “Type” or “

```
<element name="someASN1Keywords" type="x680:KeywordsSetType"/>
<complexType name="KeywordsSetType">
    <sequence maxOccurs="unbounded">
        <element name="string" type="string"/>
    </sequence>
</complexType>
```

4.10.7 Choice

tML schema supports definition of *choice* groups, which serve the same purpose as ASN.1 *choice* types.

In the interest of simplifying the implementation of tML-based TMN standards, this guideline recommends the conservative use of choice groups. Often when translating from ASN.1 to tML schema, the translated type can be simplified with no loss of semantics. For example, usually a choice between a string and null can simply be translated to a string. A comment that the string may possibly be null can be added to identify this possibility. A choice between a sequence of (or set of) and null can likewise be translated to just the sequence.

4.10.8 Object Identifier (OID)

tML schema defines a type called ID that can be used as a replacement for ASN.1 OIDs.

4.10.9 Object Instance

The ID type can also be used as in OIDs.

4.10.10 ANY

The “ANY” notation which was defined in CCITT Rec. X.208 | ISO/IEC 8824 was an open type notation. The semantics conveyed by the type ANY can be obtain through a complex type that allows any content and that does not validate the content. The following tML type definition correspond to the semantics expressed by ANY:

```
<complexType name="ANY">
    <complexContent mixed="true">
        <restriction base="anyType">
            <sequence>
                <any processContents="lax" minOccurs="0" maxOccurs="unbounded" namespace="##any"/>
            </sequence>
        </restriction>
    </complexContent>
</complexType>
```

4.10.11Tagged Types

tML schema supports isomorphic types through the use of empty extension or restriction mechanisms in complexTypes or simpleTypes respectively. However, it most be noted that tML schema does not use tags nor requires registering tags. Empty extensions of a type create another type with the same content model as the base type.

4.10.11.1 Selection Types

tML schema supports referencing named types. In order to achieve the same goal of ASN.1 selection types, define a named type for those reused types and have them qualified with the name of the super type in which they are contained. For instance,

ASN.1 definition,

```
FileAttribute ::= CHOICE {
        date-last-used      INTEGER,
        file-name VisibleString
}

AttributeList ::= SEQUENCE {
        first-attribute     date-last-used      < FileAttribute,
        second-attribute    file-name < FileAttribute
}
```

with a possible value notation of:

```
listOfAttributes AttributeList ::= {
        first-attribute     27,
        second-attribute    "PROGRAM"
}
```

tML schema declarations and definitions,

```
<simpleType name="date-last-used_FileAttribute_Type">
    <restriction base="integer"/>
</simpleType>
<simpleType name="fileName_FileAttribute_Type">
    <restriction base="string"/>
</simpleType>
<complexType name="FileAttributeType">
    <choice>
        <element name="date-last-used" type="x680:date-last-used_FileAttribute_Type"/>
        <element name="fileName" type="x680:fileName_FileAttribute_Type"/>
    </choice>
</complexType>
<complexType name="AttributeListType">
    <sequence>
        <element name="firstAttribute" type="x680:date-last-used_FileAttribute_Type"/>
        <element name="secondAttribute" type="x680:fileName_FileAttribute_Type"/>
    </sequence>
</complexType>

<element name="listOfAttributes" type="x.680:AttributeListType"/>
```

with a possible instance document:

```
<listOfAttributes>
    <firstAttribute>27</firstAttribute>
    <secondAttribute>PROGRAM</secondAttribute>
</listOfAttributes>
```

4.10.12 Bit String

This subclause defines two mappings of ASN.1 BIT STRING for use when translating GDMO specification to tML schema types, or for specifying syntaxes for use in new tML schemas.

Two mappings are justified, since ASN.1 bit strings are used in different ways.

```
BitStringType ::=     BIT STRING |
                      BIT STRING {NamedBitList}
```

The two mappings are:

- ASN.1 BIT STRING maps to a tML restricted-string of 0s and 1s ended followed by 'B'. No semantic tags are carried in the tML schema.
- ASN.1 BIT STRING is mapped to a state representation of sequence of bit elements. Local helper functions to manipulate values, and associated semantic tag constants shall be provided by the computational objects and shall not be defined nor carried by this specification.

4.10.12.1 Simple BitStringValueType Representation for ASN.1 BIT STRING

ASN.1 BIT STRING state representation is mapped to a tML schema type definition of BitStringValueType defined as a "([01]+B|[0-9A-F]+H)?" pattern. The following declaration must be included in the tML-Base (or the tML-X.680) namespace, for use with simple bit string representations:

```
<simpleType name="BitStringValueType">
    <restriction base="string">
        <pattern value="([01]+B|[0-9A-F]+H)?"/>
    </restriction>
</simpleType>
```

Because of tML schema's limitations and simplicity, the semantics for the SIZE of a bit string are limited to the higher bound plus one (for including the string type 'B' for hstring).

The interpretation of a BitStringValueType is defined as follows to match the BER encoding of BIT STRING. The bits in the bitstring, following the notation "first bit" and ""trailing bit" is specified in ISO 8824, shall be mapped to characters in the BitStringValueType content model. If the bitstring is empty, the BitStringValueType content will be empty.

ASN.1 Bit string constants are represented as a BitStringValueType content.

Table 1. Example of simple ASN.1 BIT STRING Mapping

ASN.1	**tML Schema**
BitField ::= BIT STRING (SIZE(12)) map1 BitField ::= '100110100100'B map2 BitField ::= '9A4'H	<simpleType name="BitFieldType"> <restriction base=" tML-Base:BitStringValueType "> <maxLength value="13"> </restriction> </simpleType> <element name="map1" type="BitStringValueType" fixed="100110100100B"/> <element name="map2" type="BitStringValueType" fixed="9A4H"/>
CCDScan ::= BIT STRING scan CCDScan ::= '100110100100001110110'B	<simpleType name="CCDScan"> <restriction base=" tML-Base:BitStringValueType "/> </simpleType> <element name="scan" type="CCDScan" fixed="100110100100001110110B"/>

G3FacsimilePage ::= BIT STRING -- a seq of bits conforming to Rec. T.4 image G3FacsimilePage ::= '100110100100001110110'B trailer BIT STRING ::= '01'H	<simpleType name="G3FacsimilePageType"> <restriction base="tML-Base:BitStringValueType"/> </simpleType> <!-- BitStringValue constants created for BIT STRING constants --> <element name="image" type="G3FacsimilePageType" fixed="100110100100001110110B"/> <element name="trailer" type=" tML-Base:BitStringValueType " fixed="01H"/>

4.10.12.2 BitStringType representation of ASN.1 BIT STRING

The state representation for the ASN.1 Bit String will make use of BitStringType. The representation of named bits needs to be done through a sequence of optional elements. Helper methods must be implemented in the agent and manager side applications as to understand the content of the bitstring.

```
<complexType name="BitStringType">
    <choice minOccurs="0">
        <element name="bitStringValue" type="tML-Base:BitStringValueType"/>
        <element name="bitStringNamed"/>
    </choice>
</complexType>
```

Each named bit is mapped as a tML schema element of type string with fixed value equal to the offset into the bit string. These elements will be defined within a complex element named as the bitString.

Table 2. Example Mappings to Bit String Type

ASN.1	**tML schema**
MessageFlag ::= *BIT STRING* { posResp (0), negResp (1), doNotForward (2) }	<complexType name="MessageFlagType"> <complexContent> <restriction base=" tML-Base:*BitStringType*"> <choice minOccurs="0"> <element name="bitStringValue"> <simpleType> <restriction base="tML-Base:BitStringValueType"> <maxLength value="**4**"/> </restriction> </simpleType> </element> <element name="bitStringNamed"> <complexType> <sequence> <element name="posResp" type="string" minOccurs="0" fixed="0"/> <element name="negResp" type="string" minOccurs="0" fixed="1"/> <element name="doNotForward" type="string" minOccurs="0" fixed="2"/> </sequence> </complexType> </element> </choice> </restriction> </complexContent> </complexType>
DaysOfTheWeek ::= *BIT STRING* { sunday(0), monday(1), tuesday(2), wednesday(3), thursday(4), friday(5), saturday(6) } (SIZE (0..**7**))	<complexType name="DaysOfTheWeekType"> <complexContent> <restriction base="tML-Base:*BitStringType*"> <choice minOccurs="0"> <element name="bitStringValue"> <simpleType> <restriction base="tML-Base:BitStringValueType"> <maxLength value="**8**"/> </restriction> </simpleType> </element> <element name="bitStringNamed">

sunnyDays DaysOfTheWeek ::= {sunday, monday, wednesday} cloudyDays DaysOfTheWeek ::= ‘0000010’B	<complexType> <sequence> <element name="sunday" type="string" minOccurs="0" fixed="0"/> <element name="monday" type="string" minOccurs="0" fixed="1"/> <element name="tuesday" type="string" minOccurs="0" fixed="2"/> <element name="wednesday" type="string" minOccurs="0" fixed="3"/> <element name="thursday" type="string" minOccurs="0" fixed="4"/> <element name="friday" type="string" minOccurs="0" fixed="5"/> <element name="saturday" type="string" minOccurs="0" fixed="6"/> </sequence> </complexType> </element> </choice> </restriction> </complexContent> </complexType> <element name="sunnyDays" type="DaysOfTheWeekType"/> <element name="cloudyDays" type="DaysOfTheWeekType"/> <!-- Sample tML instance documents follow --> <sunnyDays> <bitStringNamed> <sunday/> <monday/> <wednesday/> </bitStringNamed> </sunnyDays> <cloudyDays> <bitStringValue>0000010B</bitStringValue> </cloudyDays>
MessageFlag ::= *BIT STRING* { posResp (0), negResp (1), doNotForward (2) }	<complexType name="MessageFlagType"> <complexContent> <restriction base=" tML-Base:*BitStringType*"> <choice minOccurs="0"> <element name="bitStringValue"> <simpleType> <restriction base="tML-Base:BitStringValueType"> <maxLength value="**4**"/> </restriction> </simpleType> </element> <element name="bitStringNamed"> <complexType> <sequence> <element name="posResp" type="string" minOccurs="0" fixed="0"/> <element name="negResp" type="string" minOccurs="0" fixed="1"/> <element name="doNotForward" type="string" minOccurs="0" fixed="2"/> </sequence> </complexType> </element> </choice> </restriction> </complexContent> </complexType>

The mapping of complex uses of bit strings that include constants need to be further studied. Meanwhile, the modeler will have to devise ways to model it.

5 Style Idioms for tML Specifications

This section defines a set of style idioms for the tML schema to be used in mapped GDMO/ASN.1 specifications. Having a set of style idioms will result in tML schema specifications with a consistent style. This may require some additional work by editors, but this extra effort is worth the increased readability of the tML specifications. It is important to keep in perspective that style conventions are for the benefit of the reader, not necessarily to the benefit of the author.

5.1 Use Consistent Indentation

This section demonstrates the indentation style that may be used in the tML schemas. As an example, an excerpt from the tML BitStringType definition is shown below:

```
<complexType name="BitStringSequenceType">
    <sequence>
        <choice>
            <element name="bitStringValue"  type="tML-Base:BitStringType"  minOccurs="0"/>
            <element name="bitStringNamed" type="tML-Base:BSDummySetType" minOccurs="0"/>
        </choice>
        <element name="bitStringType" type="tML-Base:bitstringtypeType"/>
    </sequence>
</complexType>
```

5.2 Use Consistent Case for Identifiers

Several languages enforce case rules (such as ASN.1) while others have de-facto rules. These rules allow readers to easily distinguish identifiers of different type leading to increased readability. TML schema does enforce case, so the following rules are proposed.

- Type declarations shall have every embedded word capitalized except for the first word capitalized.
- All others (e.g., elements and entities) shall have the first letter of every embedded word capitalized.

5.3 Decouple types from elements

Whenever a possible, create an element separated from the definition of its type and reuse types already defined. See the example:

```
<element name="DaysOfTheWeek" type="DaysOfTheWeekType"/>
<complexType name="DaysOfTheWeekType">
    <!-- etc -->
</complexType>

<!-- tML instance document -->
<sunnyDays>
    <bitStringNamed>
        <sunday/>
        <monday/>
        <wednesday/>
    </bitStringNamed>
</sunnyDays>
```

5.4 Use a Consistent Type Suffix

Append the suffix “Type” to all tML schema derived types. This idiom increases readability by clearly separating type identifiers from other identifiers.

5.5 Use a Consistent Suffix for Sequence Types.

For sequences (ordered, duplicates allowed) use a suffix of “SeqType” to distinguish sequences from singulars.

5.6 Use a Consistent Suffix for Set Types.

For sets (unordered, duplicates disallowed) use a suffix of “SetType” to distinguish sets from singulars.

5.7 Use a Consistent Suffix for Optional Types

For optional types use a suffix of “TypeOpt” to distinguish them from the non-optional type.

5.8 Assume No Global Identifier Spaces

To reduce name collisions and promote reuse, all identifiers shall be scoped to a particular context (e.g., module, and interface).

5.9 Global Level Definitions

All named type and exportable element definitions shall be at the global level. Nested type definitions or local types shall be unnamed. Avert cluttering the global space with unnecessary type and element definitions.

5.10 Explicit vs. Implicit namespaces

In order to ease creation of tML schemas and readability, have qualified element namespace prefix.

6 Application to the configuration audit support function

This section deals with how the GDMO/ASN.1 mappings to XML schema, defined in the previous sections, could be applied to configuration audit support. ITU-T Recommendation X.792 is concerned with support of the auditing of the configuration of managed objects by providing for the ability of a managing system to request a managed system to request the managed system to generate a file containing the managed object information (a.k.a managed information base –MIB–).

Recommendation X.792 defines a support function that allows a manager to obtain the current view of the managed system MIB as a file. This file contains ASN.1 encoded data based on ASN.1 definitions. The scope of this proposed standard is to map the ASN.1 definitions used for generating the MIB file into an XML schema that could be used for generating a MIB file in XML format.

Like Recommendation X.792, this proposed standard does not specify how the managing system uses the information in the file once it is transferred nor the transfer mechanisms and protocols used for transferring the MIB file from the managed system to the managing system. This proposed standard only defines the approach for mapping the format of the file to an XML syntax and suggests schema for allowing the managing system to request a managed system to create the MIB file.

6.1 X.792 ASN.1 definitions and corresponding tML Schemas

The definitions contained in X.792 have two ASN.1 modules: Audit ASN1FileModule and AuditASN1Module. The former describes the syntactical stricture of the generated MIB file.
The former describes the required types for the interaction between a manager and an managed system.

6.1.1 AuditASN1FileModule

As per X.792, AuditASN1FileModule is defined as:

```
AuditASN1FileModule {itu-t recommendation x(22) configAud(792) informationModel(0) fileDefinition(3)}
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
        -- EXPORT Everything

        IMPORTS
                ObjectClass, ObjectInstance, Attribute FROM CMIP-1 {joint-iso-ccitt ms(9) cmip(1) modules(0) protocol(3)};

        AuditFile ::= SET OF SEQUENCE {
                managedObjectClass      ObjectClass OPTIONAL,
                managedObjectInstance   ObjectInstance,
                attributeList           SET OF Attribute
        }

END -- of AuditASN1FileModule
```

The corresponding schema for **InformationFramework** follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="http://www.itu.int/tML/tML-ITU-T-AuditASN1FileModule"
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    xmlns:tML-CMIP-1="http://www.itu.int/tML/tML-ITU-T-CMIP-1"
    xmlns:tML-AuditASN1FileModule="http://www.itu.int/tML/tML-ITU-T-AuditASN1FileModule"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="1.1 2001/05/04 14:22:09"
    id="ITU-T-X.792-1999-AuditASN1FileModule.xsd"
>
    <import namespace="http://www.itu.int/tML/tML-ITU-T-CMIP-1" schemaLocation="tML-X.711-CMIP-1.xsd"/>

    <annotation>
        <documentation xml:lang="en">
            Based on ITU-T X.792 (1999).
        </documentation>
    </annotation>

    <complexType name="AuditFileSetType">
        <sequence maxOccurs="unbounded">
            <element name="managedObjectClass" type="tML-CMIP-1:ObjectClassType" minOccurs="0"/>
            <element name="managedObjectInstance" type="tML-CMIP-1:ObjectInstanceType"/>
            <element name="attributeList">
                <complexType>
                    <sequence maxOccurs="unbounded">
                        <element name="Attribute" type="tML-CMIP-1:AttributeType"/>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </complexType>
</schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="http://www.itu.int/tML/tML-ITU-T-AuditASN1FileModule"
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    xmlns:tML-AuditASN1FileModule="http://www.itu.int/tML/tML-ITU-T-AuditASN1FileModule"
    xmlns:tML-Base="http://www.itu.int/tML/tML-Base"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="1.1 2001/05/04 14:22:09"
    id="tML-ITU-T-AuditASN1FileModule.xsd"
>
    <annotation>
        <documentation xml:lang="en">
            - Incorporates all the ITU-T X.792 Version 1999 Type declarations
        </documentation>
    </annotation>
    <include schemaLocation="ITU-T-X.792-1999-AuditASN1FileModule.xsd"/>
</schema>
```

The above ASN.1 module states that it imports type definitions from CMIP-1, which is an ASN.1 module defined in X.711. The productions from X.711 needed by X.792 are shown below:

```
--    Common Management Information Protocol (CMIP)

CMIP-1 {joint-iso-itu-t ms(9) cmip(1) modules(0) protocol(3)}

DEFINITIONS        ::= BEGIN
        …
        IMPORTS
        …
                -- Directory Service definitions
                -- This Recommendation | International Standard imports abstract syntax from CCI TT Rec. X.501 (1988) |
                -- ISO/IEC 9594-2:1990. Annex D to this Recommendation | International Standard provides an extract
                -- from CCITT Rec. X.501 (1988) | ISO/IEC 9594-2:1990, sufficient to meet the needs of CMIP.
                DistinguishedName, RDNSequence
                        FROM InformationFramework {joint-iso-ccitt ds(5) modules(1) informationFramework(1)};
        …

         Attribute ::= SEQUENCE {
                attributeId     AttributeId,
                attributeValue          ANY DEFINED BY          attributeId
         }
        …

         AttributeId     ::= CHOICE {
                globalForm              [0] IMPLICIT OBJECT IDENTIFIER,
                localForm               [1] IMPLICIT INTEGER
         }
        …

         ObjectClass     ::= CHOICE {
                globalForm              [0] IMPLICIT OBJECT IDENTIFIER,
                localForm               [1] IMPLICIT INTEGER
         }
         --     This Recommendation | International Standard does not allocate any values for localForm. Where this alternative is
         --     used, the permissible values or the integers and their meanings shall be defined as part of the application context in
         --     which they are used

         ObjectInstance    ::= CHOICE {
                distinguishedName                       [2] IMPLICIT DistinguishedName,
                nonSpecificForm                         [3] IMPLICIT OCTET STRING,
                localDistinguishedName                  [4] IMPLICIT RDNSequence
         }
        …
END -- End of CMIP syntax definitions
```

The corresponding schema for **InformationFramework** follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="http://www.itu.int/tML/tML-ITU-T-CMIP-1"
    xmlns:tML-CMIP-1="http://www.itu.int/tML/tML-ITU-T-CMIP-1"
    xmlns:tML-InformationFramework="http://www.itu.int/tML/tML-ITU-T-InformationFramework"
    xmlns:tML-X.680="http://www.itu.int/tML/tML-X.680"
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="1.1 2001/05/04 14:22:09"
    id="ITU-T-X.711-1997-CMIP-1.xsd"
>
    <import  namespace="http://www.itu.int/tML/tML-X.680" schemaLocation="tML-X.680.xsd"/>
    <import namespace="http://www.itu.int/tML/tML-ITU-T-InformationFramework"
            schemaLocation="tML-X.501-InformationFramework.xsd"/>

    <annotation>
        <documentation xml:lang="en">
            Based on ITU-T X.711 (1997 E).
        </documentation>
    </annotation>

    <complexType name="AttributeType">
        <sequence>
            <element name="attributeId" type="tML-CMIP-1:AttributeIdType"/>
            <element name="attributeValue" type="tML-X.680:ANY"/>
        </sequence>
    </complexType>
    <complexType name="AttributeIdType">
        <choice>
            <element name="globalForm" type="ID"/>
            <element name="localForm" type="integer"/>
        </choice>
    </complexType>
    <complexType name="ObjectClassType">
        <choice>
            <element name="globalForm" type="ID"/>
            <element name="localForm" type="integer"/>
        </choice>
    </complexType>
    <complexType name="ObjectInstanceType">
        <choice>
            <element name="distinguishedName" type="tML-InformationFramework:DistinguishedNameType"/>
            <element name="nonSpecific" type="binary"/>
            <element name="localDistinguishedName" type="tML-InformationFramework:RDNSequenceType"/>
        </choice>
    </complexType>
</schema>

<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.itu.int/tML/tML-ITU-T-CMIP-1" xmlns:tML-CMIP-1="http://www.itu.int/tML/tML-ITU-T-CMIP-1"
xmlns="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.1
2001/05/04 14:22:09" id="tML-X.711-CMIP-1.xsd">
    <annotation>
        <documentation xml:lang="en">
            - Incorporates all the ANSI X.711 Version 1997 CMIP-1Type declarations
        </documentation>
    </annotation>
    <include schemaLocation="ITU-T-X.711-1997-CMIP-1.xsd"/>
</schema>
```

The above ASN.1 module states that it imports type definitions from InformationFramework, which is an ASN.1 module defined in X.501. The productions from X.501 needed by X.711 are shown below:

```
InformationFramework {joint-iso-itu-t ds(5) module(1) informationFramework(1) 3}
DEFINITIONS ::=
BEGIN
        …
        AttributeType ::= OBJECT IDENTIFIER
        AttributeValue ::= ANY
        AttributeValueAssertion ::= SEQUENCE {
                type AttributeType,
                assertion AttributeValue
        }
        Name ::= CHOICE { rdnSequence RDNSequence }
        RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
        DistinguishedName ::= RDNSequence
        RelativeDistinguishedName ::= SET OF AttributeValueAssertion
        …
END
```

The corresponding schema for **InformationFramework** follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="http://www.itu.int/tML/tML-ITU-T-InformationFramework"
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    xmlns:tML-InformationFramework="http://www.itu.int/tML/tML-ITU-T-InformationFramework"
    xmlns:tML-X.680="http://www.itu.int/tML/tML-X.680"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="1.1 2001/05/04 14:22:09"
    id="ITU-T-X.501-1997-InformationFramework.xsd"
>
    <import namespace="http://www.itu.int/tML/tML-X.680" schemaLocation="tML-X.680.xsd"/>
    <annotation>
        <documentation xml:lang="en">
            Based on ITU-T X.501 (1997) InformationFramework Module.
        </documentation>
    </annotation>
    <!-- etc -->
    <simpleType name="AttributeType Type">
        <restriction base="ID"/>
    </simpleType>
    <complexType name="AttributeValue Type">
        <complexContent>
            <extension base="tML-X.680:ANY "/>
        </complexContent>
    </complexType>
    <complexType name="AttributeValueAssertion Type">
        <sequence>
            <element name="type" type="tML-InformationFramework:AttributeTypeType"/>
            <element name="assertion" type="tML-InformationFramework:AttributeValueType"/>
        </sequence>
    </complexType>
    <complexType name="Name Type">
        <choice>
            <element name="rdnSequence" type="tML-InformationFramework:RDNSequenceType"/>
        </choice>
    </complexType>
    <complexType name="RDNSequence Type">
        <sequence>
            <element name="type" type="tML-InformationFramework:RelativeDistinguishedNameSetType"
                                    maxOccurs="unbounded"/>
        </sequence>
    </complexType>
```

```
    <complexType name="DistinguishedNameType">
        <complexContent>
            <extension base="tML-InformationFramework:RDNSequenceType"/>
        </complexContent>
    </complexType>
    <complexType name="RelativeDistinguishedNameSetType">
        <sequence>
            <element name="type" type="tML-InformationFramework:AttributeValueAssertionType" maxOccurs="unbounded"/>
        </sequence>
    </complexType>
    <!-- etc -->
</schema>

<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="http://www.itu.int/tML/tML-ITU-T-InformationFramework"
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    xmlns:tML-InformationFramework="http://www.itu.int/tML/tML-ITU-T-InformationFramework"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="1.1 2001/05/04 14:22:09"
    id="tML-X.501-InformationFramework.xsd"
>
    <annotation>
        <documentation xml:lang="en">
            - Incorporates all the ITU-T X.501 Version 1997 InformationFrameworkType declarations
        </documentation>
    </annotation>

    <include schemaLocation="ITU-T-X.501-1997-InformationFramework.xsd"/>

</schema>
```

A sample instance tML document corresponding to a sample MIB file follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<AuditFile
    xmlns="http://www.itu.int/tML/tML-ITU-T-AuditASN1FileModule"
    xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
    xmlns:cmip="http://www.itu.int/tML/tML-ITU-T-CMIP-1"
    xmlns:ds="http://www.itu.int/tML/tML-ITU-T-InformationFramework"
    xsi:schemaLocation="http://www.itu.int/tML/tML-ITU-T-AuditASN1FileModule ITU-T-X.792-1999-AuditASN1FileModule.xsd"
>
    <managedObjectInstance>
        <cmip:distinguishedName>
            <ds:RDN>
                <ds:AttributeValueAssertion>
                    <ds:type>OBJECT12345</ds:type>
                    <ds:assertion>anyAssertion</ds:assertion>
                </ds:AttributeValueAssertion>
            </ds:RDN>
        </cmip:distinguishedName>
    </managedObjectInstance>
    <attributeList>
        <Attribute>
            <cmip:attributeId>
                <cmip:globalForm>ATTRIBUTE54321</cmip:globalForm>
            </cmip:attributeId>
            <cmip:attributeValue>1f2e23da</cmip:attributeValue>
        </Attribute>
    </attributeList>
</AuditFile>
```

6.1.2 AuditASN1Module

As per X.792, AuditASN1Module is defined as:

```
AuditASN1Module
        {itu-t recommendation x(22) configAud(792) informationModel(0) asn1Modules(2) asn1DefinedTypeModules(0)}
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
        -- EXPORTS everything

        IMPORTS  NameType FROM
                        ASN1DefinedTypesModule
                        {ccitt recommendation m gnm(3100) informationModel(0) asn1Modules(2) asn1DefinedTypesModule(0)} ;

        configAuditSup OBJECT IDENTIFIER ::= {itu-t recommendation x configAud(752) informationModel(0) }

        configAuditSupstandardSpecificExtension OBJECT IDENTIFIER ::= {configAuditSup standardSpecificExtension(0)}

        configAuditSupMObjectClass OBJECT IDENTIFIER ::= {configAuditSup managedObjectClass(3)}

        configAuditSupPackage OBJECT IDENTIFIER ::= {configAuditSup package(4)}

        configAuditSupParameter OBJECT IDENTIFIER ::= {configAuditSup parameter(5)}

        configAuditSupAttribute OBJECT IDENTIFIER ::= {configAuditSup attribute(7)}

        configAuditSupNameBinding OBJECT IDENTIFIER ::= {configAuditSup nameBinding(6)}

        configAuditSupAction OBJECT IDENTIFIER ::= {configAuditSup action(9)}

        configAuditSupNotification OBJECT IDENTIFIER ::= {configAuditSup notification(10)}
        -- Reserved arcs below configAuditSup are (8) for attribute groups

        AuditInitiateInformation ::= SEQUENCE {
                                        classes SEQUENCE {
                                                include CHOICE {
                                                                all NULL,
                                                                subset SET OF OBJECT IDENTIFIER
                                                },
                                                exclude CHOICE {
                                                                none NULL,
                                                                subset SET OF OBJECT IDENTIFIER
                                                }
                                        },
                                        attributes SEQUENCE {
                                                include CHOICE {
                                                                all NULL,
                                                                subset SET OF OBJECT IDENTIFIER
                                                },
                                                exclude CHOICE {
                                                                none NULL,
                                                                subset SET OF OBJECT IDENTIFIER
                                                }
                                        }
        }

        AuditInitiateResult ::= ENUMERATED
                { started(0) auditAlreadyRunning(1) resourceLimit(2) notStartedOther(3) }

        AuditTerminateResult ::= ENUMERATED
                { terminated(0) noAuditRunning(1) couldNotTerminate(2) }
```

```
AuditResult ::= ENUMERATED
        { succeeded(0) partialSuccess(1) terminatedExternally(2) terminatedInternally(3) terminatedNoFileSpace(4) }

FileSuspect ::= BOOLEAN -- False means file is OK (not suspect)

OutputResult ::= SEQUENCE OF SEQUENCE {
                        filename GraphicString,
                        fileSuspect FileSuspect DEFAULT FALSE
}

AuditReportInformation ::= SEQUENCE {
                result          AuditResult,
                outputResult    OutputResult    OPTIONAL
}

Boolean ::= BOOLEAN

TimeStamp ::= CHOICE {
        neverRun NULL,
        run             GeneralizedTime
}

END     -- of AuditASN1Module
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="http://www.itu.int/tML/tML-ITU-T-AuditASN1Module"
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    xmlns:tML-CMIP-1="http://www.itu.int/tML/tML-ITU-T-CMIP-1"
    xmlns:tML-AuditASN1Module="http://www.itu.int/tML/tML-ITU-T-AuditASN1Module"
    xmlns:tML-ASN1DefinedTypesModule="http://www.itu.int/tML/tML-ITU-T-ASN1DefinedTypesModule"
    xmlns:tML-X.680="http://www.itu.int/tML/tML-X.680"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="1.1 2001/05/04 14:22:09"
    id="ITU-T-X.792-1999-AuditASN1Module.xsd"
>
    <annotation>
        <documentation xml:lang="en">
            Based on ITU-T  X.792 (1999).
        </documentation>
    </annotation>
    <import namespace="http://www.itu.int/tML/tML-ITU-T-CMIP-1" schemaLocation="tML-X.711-CMIP-1.xsd"/>
    <import namespace="tML-ITU-T-ASN1DefinedTypesModule" schemaLocation="tML-M.3100-ASN1DefinedTypesModule.xsd"/>
    <import namespace="http://www.itu.int/tML/tML-X.680" schemaLocation="tML-X.680.xsd"/>

    <simpleType name="configAuditSupType">
        <restriction base="ID"/>
    </simpleType>
    <simpleType name="configAuditSupstandardSpecificExtensionType">
        <restriction base="ID"/>
    </simpleType>
    <simpleType name="configAuditSupMObjectClassType">
        <restriction base="ID"/>
    </simpleType>
    <simpleType name="configAuditSupPackageType">
        <restriction base="ID"/>
    </simpleType>
    <simpleType name="configAuditSupParameterType">
        <restriction base="ID"/>
    </simpleType>
    <simpleType name="configAuditSupAttributeType">
        <restriction base="ID"/>
    </simpleType>
```

```
<simpleType name="configAuditSupNameBindingType">
    <restriction base="ID"/>
</simpleType>
<simpleType name="configAuditSupActionType">
    <restriction base="ID"/>
</simpleType>
<simpleType name="configAuditSupNotificationType">
    <restriction base="ID"/>
</simpleType>
<complexType name="AuditInitiateInformationType">
    <sequence>
        <element name="classes">
            <complexType>
                <sequence maxOccurs="unbounded">
                    <element name="Attribute" type="tML-CMIP-1:AttributeType"/>
                </sequence>
            </complexType>
        </element>
        <element name="managedObjectInstance" type="tML-CMIP-1:ObjectInstanceType"/>
        <element name="attributeList">
            <complexType>
                <sequence maxOccurs="unbounded">
                    <element name="Attribute" type="tML-CMIP-1:AttributeType"/>
                </sequence>
            </complexType>
        </element>
    </sequence>
</complexType>
<complexType name="subsetSetType">
    <sequence maxOccurs="unbounded">
        <element name="ObjectIdentifier" type="ID"/>
    </sequence>
</complexType>
<complexType name="includeType">
    <choice>
        <element name="all" type="tML-X.680:NULL"/>
        <element name="subset" type="tML-AuditASN1Module:subsetSetType"/>
    </choice>
</complexType>
<complexType name="excludeType">
    <choice>
        <element name="none" type="tML-X.680:NULL"/>
        <element name="subset" type="tML-AuditASN1Module:subsetSetType"/>
    </choice>
</complexType>
<complexType name="includeExcludeType">
    <choice>
        <element name="include" type="tML-AuditASN1Module:includeType"/>
        <element name="exclude" type="tML-AuditASN1Module:excludeType"/>
    </choice>
</complexType>
<complexType name="AuditInitiateInformationType">
    <sequence>
        <element name="classes" type="tML-AuditASN1Module:includeExcludeType"/>
        <element name="attributes" type="tML-AuditASN1Module:includeExcludeType"/>
    </sequence>
</complexType>
<simpleType name="AuditInitiateResultType">
    <restriction base="string">
        <enumeration value="started"/>
        <enumeration value="auditAlreadyRunning"/>
        <enumeration value="resourceLimit"/>
        <enumeration value="notStartedOther"/>
    </restriction>
</simpleType>
```

```
    <simpleType name="AuditTerminateResultType">
        <restriction base="string">
            <enumeration value="terminated"/>
            <enumeration value="noAuditRunning"/>
            <enumeration value="couldNotTerminate"/>
        </restriction>
    </simpleType>
    <simpleType name="AuditResultType">
        <restriction base="string">
            <enumeration value="succeeded"/>
            <enumeration value="partialSuccess"/>
            <enumeration value="terminatedExternally"/>
            <enumeration value="terminatedInternally"/>
            <enumeration value="terminatedNoFileSpace"/>
        </restriction>
    </simpleType>
    <simpleType name="FileSuspectType">
        <annotation>
            <documentation>False means file is OK (not suspect) </documentation>
        </annotation>
        <restriction base="boolean"/>
    </simpleType>
    <complexType name="OutputResultType">
        <sequence maxOccurs="unbounded">
            <element name="filename" type="string"/>
            <element name="fileSuspect" type="tML-AuditASN1Module:FileSuspectType" default="false"/>
        </sequence>
    </complexType>
    <complexType name="AuditReportInformationType">
        <sequence maxOccurs="unbounded">
            <element name="result" type="tML-AuditASN1Module:AuditResultType"/>
            <element name="outputResult" type="tML-AuditASN1Module:OutputResultType" minOccurs="0"/>
        </sequence>
    </complexType>
    <simpleType name="BooleanType">
        <annotation>
            <documentation>!!! This is redundant !!! </documentation>
        </annotation>
        <restriction base="boolean"/>
    </simpleType>
    <complexType name="TimeStampType">
        <choice>
            <element name="neverRun" type="tML-X.680:NULL"/>
            <element name="run" type="dateTime"/>
        </choice>
    </complexType>
</schema>

<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="http://www.itu.int/tML/tML-ITU-T-AuditASN1Module"
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    xmlns:tML-AuditASN1Module="http://www.itu.int/tML/tML-ITU-T-AuditASN1Module"
    xmlns:tML-Base="http://www.itu.int/tML/tML-Base"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="1.1 2001/05/04 14:22:09"
    id="tML-ITU-T-AuditASN1Module.xsd"
>
    <annotation>
        <documentation xml:lang="en">- Incorporates all the ITU-T X.792 Version 1999 Type declarations</documentation>
    </annotation>
    <include schemaLocation="ITU-T-X.792-1999-AuditASN1Module.xsd"/>
</schema>
```

The above ASN.1 module states that it imports type definitions from ASN1DefinedTypesModule, which is an ASN.1 module defined in M.3100. The productions from M.3100needed by X.729 are shown below:

```
ASN1DefinedTypesModule
        {ccitt recommendation m gnm(3100) informationModel(0) asn1Modules(2) asn1DefinedTypesModule(0)}
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
        …

        NameType ::=   CHOICE {
                    numericName INTEGER,
                    pString         GraphicString
        }
        …

END -- end of ASN1 DefinedTypesModule
```

The corresponding schema for **ASN1DefinedTypesModule** follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="http://www.itu.int/tML/tML-ITU-T-ASN1DefinedTypesModule"
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    xmlns:tML-ASN1DefinedTypesModule="http://www.itu.int/tML/tML-ITU-T-ASN1DefinedTypesModule"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="1.1 2001/05/04 14:22:09"
    id="ITU-T-M.3100-1995-ASN1DefinedTypesModule.xsd"
>
    <annotation>
        <documentation xml:lang="en">
            Based on ITU-T M.3100 (1995) ASN1DefinedTypesModule Module.
        </documentation>
    </annotation>
    <!-- etc -->
    <complexType name="NameTypeType">
        <choice>
            <element name="numericName" type="integer"/>
            <element name="pString" type="string"/>
        </choice>
    </complexType>
    <!-- etc -->
</schema>

<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="http://www.itu.int/tML/tML-ITU-T-ASN1DefinedTypesModule"
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    xmlns:tML-ASN1DefinedTypesModule="http://www.itu.int/tML/tML-ITU-T-ASN1DefinedTypesModule"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="1.1 2001/05/04 14:22:09"
    id="tML-M.3100-ASN1DefinedTypesModule.xsd"
>
    <annotation>
        <documentation xml:lang="en">
            - Incorporates all the ANSI M.3100 Version 1997 ASN1DefinedTypesModuleType declarations
        </documentation>
    </annotation>
    <include schemaLocation="ITU-T-M.3100-1995-ASN1DefinedTypesModule.xsd"/>
</schema>
```

Annex A The BitString, ANY and NULL type definitions

(Normative)

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    targetNamespace="http://www.itu.int/tML/tML-X.680"
    xmlns:tML-X.680="http://www.itu.int/tML/tML-X.680"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
>
    <complexType name="BitStringType" abstract="true" block="extension">
        <annotation>
            <documentation xml:lang="en">
                This type has to be derived by restriction so that the "bitStringNamed" element is further defined.
                As per ITU-T X.680 (1997 E), bit strings are defined as BitStringType::=BITSTRING|BITSTRING {NamedBitList}
            </documentation>
        </annotation>
        <choice minOccurs="0">
            <element name="bitStringValue" type="tML-X.680:BitStringValueType"/>
            <element name="bitStringNamed"/>
        </choice>
    </complexType>

    <simpleType name="BitStringValueType">
        <restriction base="string">
            <pattern value="([01]+B|[0-9A-F]+H)?"/>
        </restriction>
    </simpleType>

    <complexType name="NULL" final="all"/>

    <complexType name="ANY">
        <complexContent mixed="true">
            <restriction base="anyType">
                <sequence>
                    <any processContents="lax" minOccurs="0" maxOccurs="unbounded" namespace="##any"/>
                </sequence>
            </restriction>
        </complexContent>
    </complexType>

</schema>
```

Annex B Examples and hints[4]

(This annex does not form an integral part of this recommended standard)

This annex contains examples of the use of these guidelines for mapping ASN.1 (hypothetical) data structures to tML schemas. It contains hints, or guidelines, for mapping the various features of ASN.1 to tML schemas. Unless otherwise stated, an environment of AUTOMATIC TAGS is assumed.

B.1 Example of a personnel record

The mapping of ASN.1 to tML schemas is illustrated by means of a simple, hypothetical personnel record.

B.1.1 Example of a personnel record

The mapping of ASN.1 to tML schemas is illustrated by means of a simple, hypothetical personnel record.

Name	Raymond E Reeves
Title	Member Technical Staff
Employee Number	51
Date of Hire	17 September 1971
Name of Spouse	Mary T Reeves
Number of Children	2
Child Information	
Name	Ralph T Reeves
Date of Birth	11 November 1957
Child Information	
Name	Susan B Jones
Date of Birth	17 July 1959

B.1.2 ASN.1 description and tML schema mapping of the record structure

The structure of every record is formally described bellow using the standard notation for data types. This description is then mapped to tML schema following the guidelines recommended in this document.

ASN.1 description:

```
PersonnelRecord ::= [APPLICATION 0] SET
{
  name          Name,
  title         VisibleString,
  number        EmployeeNumber,
  dateOfHire    Date,
  nameOfSpouse  Name,
  children      SEQUENCE OF ChildInformation
                            DEFAULT {}
}
ChildInformation ::= SET
{
  name          Name,
  dateOfBirth   Date
```

[4] This section resembles Annex C in ITU-T Recommendation X.680 (1997 E) | ISOC/IEC 8824-1:1998 (E), Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.

```
}
Name ::= [APPLICATION 1] SEQUENCE
{
  givenName        VisibleString,
  initial          VisibleString,
  familyName       VisibleString
}
EmployeeNumber ::= [APPLICATION 2] INTEGER
Date ::= [APPLICATION 3] VisibleString -- YYYY MMDD
```

tML schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    xmlns="http://www.w3.org/2000/10/XMLSchema"
    targetNamespace="http://www.itu.int/tML/tML-X.680"
    xmlns:x680="http://www.itu.int/tML/tML-X.680"

    elementFormDefault="qualified"
    attributeFormDefault="unqualified"

    id="tML-X.680.xsd"
    version="1.0 2001/05/20 13:10:01"
>

    <element name="PersonnelRecord" type="x680:PersonnelRecordSetType"/>

    <complexType name="PersonnelRecordSetType">
        <sequence>
            <element name="name" type="x680:NameType"/>
            <element name="title" type="string"/>
            <element name="number" type="x680:EmployeeNumberType"/>
            <element name="dateOfHire" type="x680:DateType"/>
            <element name="nameOfSpouse" type="x680:NameType"/>
            <element name="children">
                <complexType>
                    <sequence>
                        <element name="ChildInformation" type="x680:ChildInformationSetType"
                                 minOccurs="0" maxOccurs="unbounded"/>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </complexType>

    <complexType name="ChildInformationSetType">
        <sequence>
            <element name="name" type="x680:NameType"/>
            <element name="dateOfBirth" type="x680:DateType"/>
        </sequence>
    </complexType>

    <complexType name="NameType">
        <sequence>
            <element name="givenName" type="string"/>
            <element name="initial" type="string"/>
            <element name="familyName" type="string"/>
        </sequence>
    </complexType>

    <simpleType name="EmployeeNumberType">
        <restriction base="integer"/>
    </simpleType>

    <!-- This one would have been created automatically
```

```
        <simpleType name=" DateType"><restriction base="string"/></simpleType>

        However, because manual refinements are done, the next declaration is used:
        -->
        <simpleType name=" DateType"><restriction base="date"/></simpleType>

    </schema>
```

This example illustrates an aspect of the mapping from ASN.1 to tML. Set types are mapped to sequence types whose names are ending with “SetType”. The syntactic construct “DEFAULT {}” applied to “children” is translated into a minOccurs="0" for the ChildInformation element.

B.1.3 ASN.1 description of record value and corresponding tML instance document

The value of Raymond E Reeves’ personnel record is formally described below using the standard ASN.1 notation for data value.

```
{
        name            {givenName "Raymond", initial "E", familyName "Reeves"},
        title           "Member Technical Staff",
        number          51,
        dateOfHire      "19710917",
        nameOfSpouse    {givenName "Mary", initial "T", familyName " Reeves"},
        children
        {
                {
                        name {givenName "Ralph", initial "T", familyName " Reeves"} ,
                        dateOfBirth "19571111"
                },
                {
                        name {givenName "Susan", initial "B", familyName "Jones"} ,
                        dateOfBirth "19590717"
                }
        }
}
```

The corresponding tML instance document is:

```
    <?xml version="1.0" encoding="UTF-8"?>
    <PersonnelRecord
        xmlns="http://www.itu.int/tML/tML-X.680"
        xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
        xsi:schemaLocation="http://www.itu.int/tML/tML-X.680 tML-X.680.xsd"
    >
        <name>
            <givenName>Raymond</givenName>
            <initial>E</initial>
            <familyName>Reeves</familyName>
        </name>
        <title>Member Technical Staff</title>
        <number>51</number>
        <dateOfHire>1971-09-17</dateOfHire>
        <nameOfSpouse>
            <givenName>Mary</givenName>
            <initial>T</initial>
            <familyName>Reeves</familyName>
        </nameOfSpouse>
        <children>
            <ChildInformation>
                <name>
                    <givenName>Ralph</givenName>
                    <initial>T</initial>
                    <familyName>Reeves</familyName>
                </name>
                <dateOfBirth>1957-11-11</dateOfBirth>
```

```
        </ChildInformation>
        <ChildInformation>
            <name>
                <givenName>Susan</givenName>
                <initial>B</initial>
                <familyName>Jones</familyName>
            </name>
            <dateOfBirth>1959-07-17</dateOfBirth>
        </ChildInformation>
    </children>
</PersonnelRecord>
```

B.2 Examples for data types in general

ASN.1	tML Schema

Employed ::= *BOOLEAN*	`<element name="Employed" type="boolean"/>`
	or…
	`<element name="Employed" type="x680:EmployedType"/>`
	`<simpleType name="EmployedType">`
	`  <restriction base="boolean"/>`
	`</simpleType>`

CheckingAccountBalance ::= *INTEGER*
-- in cents; negative means overdrawn.
	`<element name="balance" type="x680:CheckingAccountBalanceType"/>`
	`<simpleType name="CheckingAccountBalanceType">`
	`  <restriction base="integer"/>`
	`</simpleType>`

DayOfTheMonth ::= *INTEGER*
{first(1), last(31)} (first .. last)
	`<simpleType name="DayOfTheMonthType">`
	`  <restriction base="integer">`
	`  <minInclusive value="1"/>`
	`  <maxInclusive value="31"/>`
	`  </restriction>`
	`</simpleType>`

DayOfTheWeek ::= *ENUMERATED*
{
sunday(0),
monday(1),
tuesday(2),
wednesday(3),
thursday(4),
friday(5),
saturday(6)
}
	`<simpleType name="DayOfTheWeekType">`
	`  <restriction base="string">`
	`  <enumeration value="sunday"/>`
	`  <enumeration value="monday"/>`
	`  <enumeration value="tuesday"/>`
	`  <enumeration value="wednesday"/>`
	`  <enumeration value="thursday"/>`
	`  <enumeration value="friday"/>`
	`  <enumeration value="saturday"/>`
	`  </restriction>`
	`</simpleType>`

pi *REAL* ::= { **mantissa 3141592653589793238462643383279,** **base 10,** **exponent –30** **}**	`<element name="pi" type="double"` `  fixed="3141592653589793238462643383279E-30"/>`

Surname ::= *PrintableString*
president Surname ::= "Clinton"
	`<element name="president" type="x680:SurnameType" fixed="Clinton"/>`
	`<simpleType name="SurnameType">`
	`  <restriction base="string"/>`
	`</simpleType>`

PatientIdentifier ::= *SEQUENCE* {
name *VisibleString*,
roomNumber *CHOICE* {
room *INTEGER*,
outPatient *NULL* *-- if an out-patient --*
}
}
 	
lastPatient PatientIdentifier ::= {
name "Jane Doe",
roomNumber outPatient : NULL
}
	`<element name="lastPatient" type="x680:patientIdentifierType"/>`
	`<complexType name="patientIdentifierType">`
	`  <sequence>`
	`  <element name="name" type="string"/>`
	`  <element name="roomNumber">`
	`  <complexType>`
	`  <choice>`
	`  <element name="room" type="integer"/>`
	`  <element name="outPatient" type="x680:NULL"/>`
	`  </choice>`
	`  </complexType>`
	`  </element>`
	`  </sequence>`
	`</complexType>`

	<!-- Instance document --> <lastPatient> <name>Jane Doe</name> <roomNumber> <outPatient/> </roomNumber> </lastPatient> <!-- NULL type as defined in the base tML schema --> <complexType name=" NULL"/>
Credentials ::= *SEQUENCE* { **userName VisibleString,** **password VisibleString,** **accountNumber INTEGER** **}**	<complexType name="CredentialsType"> <*sequence*> <element name="userName" type="string"/> <element name="password" type="string"/> <element name="accountNumber" type="integer"/> </*sequence*> </complexType>
UserName ::= SET { **personalName [0] VisibleString,** **organizationName [1] VisibleString,** **countryName [2] VisibleString** **}**	<complexType name="UserName *SetType*"> <*sequence*> <element name="personalName" type="string"/> <element name="organizationName" type="string"/> <element name="countryName" type="string"/> </*sequence*> </complexType>
Keywords ::= SET OF VisibleString *-- in arbitrary order* **someASN1Keywords Keywords ::= {"INTEGER", "BOOLEAN", "REAL"}**	<element name="someASN1Keywords" type="x680:KeywordsSetType"/> <complexType name="KeywordsSetType"> <annotation> <documentation> -- in arbitrary order </documentation> </annotation> <sequence> <element name="string" type="string" maxOccurs="unbounded"/> </sequence> </complexType> <!-- Sample instance document --> <someASN1Keywords> <string>INTEGER</string> <string>BOOLEAN</string> <string>REAL</string> </someASN1Keywords>
FileIdentifier ::= *CHOICE* { **relativeName *VisibleString*,** *-- name of file (for example, "MarchProgressReport")* **absoluteName *VisibleString*,** *-- name of file and containing directory* *-- (for example, "<Williams>MarchProgressReport")* **serialNumber *INTEGER*** *-- system-assigned identifier for file –* **}** **file FileIdentifier ::= serialNumber : 106448503**	<complexType name="FileIdentifierType"> <*choice*> <element name="relativeName" type="*string*"/> <element name="absoluteName" type="*string*"/> <element name="serialNumber" type="*integer*"/> </choice> </complexType> <element name="file" type="x680:FileIdentifierType"/> <!-- Sample instance document --> <file> <serialNumber>106448503</serialNumber> </file>
FileAttribute ::= *CHOICE* { **date-last-used INTEGER,** **file-name VisibleString** **}** **AttributeList ::= *SEQUENCE* {** **first-attribute date-last-used < FileAttribute,** **second-attribute file-name < FileAttribute** **}** **listOfAttributes AttributeList ::= {** **first-attribute 27,**	<simpleType name="date-last-used_*FileAttribute*_Type"> <restriction base="integer"/> </simpleType> <simpleType name="fileName_*FileAttribute*_Type"> <restriction base="string"/> </simpleType> <complexType name="FileAttributeType"> <*choice*> <element name="date-last-used"

second-attribute "PROGRAM" **}**	type="x680:date-last-used_FileAttribute_Type"/> <element name="fileName" type="x680:fileName_FileAttribute_Type"/> </choice> </complexType> <complexType name="AttributeListType"> <sequence> <element name="firstAttribute" type="x680:date-last-used_FileAttribute_Type"/> <element name="secondAttribute" type="x680:fileName_FileAttribute_Type"/> </sequence> </complexType> <element name="listOfAttributes" type="x680:AttributeListType"/> <!-- Instance document --> <listOfAttributes> <firstAttribute>27</firstAttribute> <secondAttribute>PROGRAM</secondAttribute> </listOfAttributes>
AtomicNumber ::= INTEGER (1..104)	<simpleType name="AtomicNumberType"> <restriction base="integer"> <minInclusive value="1"/> <maxInclusive value="104"/> </restriction> </simpleType>
TouchToneString ::= IA5String **(FROM ("0123456789" | "*" | "#")) (SIZE (1..63))**	<simpleType name="TouchToneStringType"> <restriction base="string"> <pattern value="[\d*#]{1,63}"/> </restriction> </simpleType>
ParameterList ::= SET SIZE (1..63) OF Parameter	<complexType name="ParameterList**SetType**"> <sequence> <element name="Parameter" type="string" maxOccurs="**63**"/> </sequence> </complexType>
SmallPrime ::= INTEGER (2|3|5|7|11|13|17|19|23|29)	<element name="SmallPrime" type="x680:SmallPrimeType"/> <simpleType name="SmallPrimeType"> <restriction base="integer"> <enumeration value="2"/> <enumeration value="3"/> <enumeration value="5"/> <enumeration value="7"/> <enumeration value="11"/> <enumeration value="13"/> <enumeration value="17"/> <enumeration value="19"/> <enumeration value="23"/> <enumeration value="29"/> </restriction> </simpleType>