INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.822.1

(10/2001)

SERIES Q: SWITCHING AND SIGNALLING

Q3 interface

CORBA-based TMN performance management service

ITU-T Recommendation Q.822.1

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4 AND No. 5	Q.120–Q.249
SPECIFICATIONS OF SIGNALLING SYSTEM No. 6	Q.250–Q.309
SPECIFICATIONS OF SIGNALLING SYSTEM R1	Q.310–Q.399
SPECIFICATIONS OF SIGNALLING SYSTEM R2	Q.400–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.799
General	Q.700
Message transfer part (MTP)	Q.701–Q.709
Signalling connection control part (SCCP)	Q.711–Q.719
Telephone user part (TUP)	Q.720–Q.729
ISDN supplementary services	Q.730–Q.739
Data user part	Q.740–Q.749
Signalling System No. 7 management	Q.750–Q.759
ISDN user part	Q.760–Q.769
Transaction capabilities application part	Q.770–Q.779
Test specification	Q.780–Q.799
Q3 INTERFACE	**Q.800–Q.849**
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
General	Q.850–Q.919
Data link layer	Q.920–Q.929
Network layer	Q.930–Q.939
User-network management	Q.940–Q.949
Stage 3 description for supplementary services using DSS1	Q.950–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1699
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000	Q.1700–Q.1799
SPECIFICATIONS OF SIGNALLING RELATED TO BEARER INDEPENDENT CALL CONTROL (BICC)	Q.1900–Q.1999
BROADBAND ISDN	Q.2000–Q.2999

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

ITU-T Recommendation Q.822.1

CORBA-based TMN performance management service

Summary

This Recommendation 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 Recommendation is to define a CORBA/IDL model similar to that defined in ITU-T Recs. X.739 and Q.822 using CMISE. This Recommendation is compliant with CORBA modelling standards in ITU-T X.780, X.780.1, Q.816, Q.816.1 and M.3120.

Source

ITU-T Recommendation Q.822.1 was prepared by ITU-T Study Group 4 (2001-2004) and approved under the WTSA Resolution 1 procedure on 7 October 2001.

NOTE

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

CONTENTS

ITU-T Recommendation Q.822.1

CORBA-based TMN performance management service

1 Scope

This Recommendation 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 in both fine-grained and facade version, data types, and constants. The intent of this Recommendation is to define a CORBA/IDL specification similar to that defined in ITU-T Recs. X.739 and Q.822 using CMISE. This Recommendation is compliant with proposed CORBA modelling standards in ITU-T Recs. X.780, X.780.1, Q.816, Q.816.1 and M.3120.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; 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. A list of the currently valid ITU-T Recommendations is regularly published.

[1] OMG Document formal/99-10-07, *The Common Object Request Broker: Architecture and Specification,* Revision 2.3.1.

[2] OMG Document formal/2000-06-20, *Notification Service Specification,* Version 1.0.

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

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

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

[6] ITU-T Recommendation X.780.1 (2001), *TMN guidelines for defining coarse-grained CORBA managed objects interfaces*.

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

[8] ITU-T Recommendation X.721 (1992), *Information technology – Open systems interconnection – Structure of management information: Definition of management information*.

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

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

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

[12] ITU-T Recommendation Q.823 (1996), *Stage 2 and stage 3 functional specifications for traffic management*.

3 Abbreviations

This Recommendation uses the following abbreviations:

CMIP	Common Management Information Protocol
CORBA	Common Object Request Broker Architecture
DN	Distinguished Name
FDN	Fully Distinguished Name
GDMO	Guidelines for the Definition of Managed Objects
NE	Network Element
PM	Performance Management
QoS	Quality of Service
RDN	Relative Distinguished Name
SNMP	Simple Network Management Protocol
TL1	Transaction Language 1
TMN	Telecommunications Management Network
UML	Unified Modelling Language

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 measurement 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 ITU-T 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 ITU-T Rec. Q.823. ITU-T Rec. Q.823 reuses the mechanisms of ITU-T Rec. Q.822 together with the generic simple scanner of ITU-T 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.

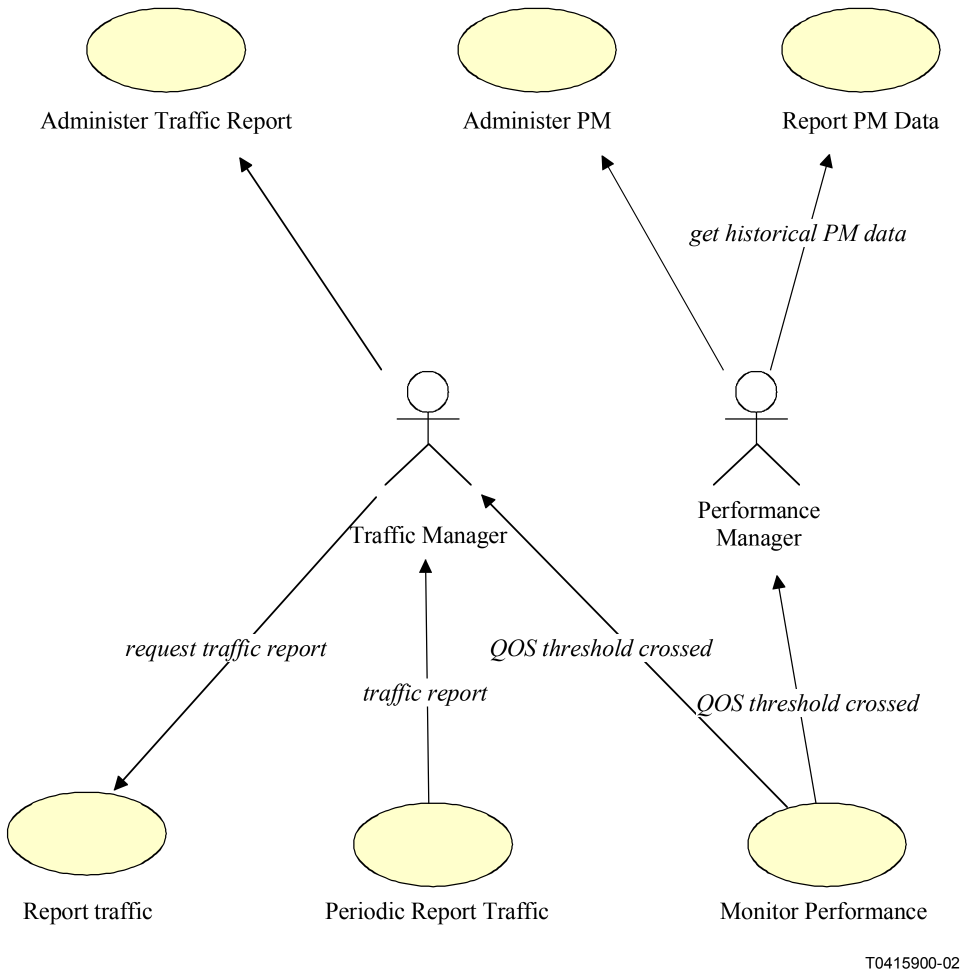


Figure 1/Q.822.1 – Use cases for performance measurement

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 Recommendation follows the information modelling guidelines defined in ITU-T Rec. X.780. In addition, this Recommendation tries to keep the model practical, simple and to make sense to users and manufacturers.

4.2 Service Modelling

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 modelling guidelines and the needs from existing and current PM applications, not all the capabilities defined for these managed objects in ITU-T Recs. 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 a potentially large number of managed objects. These entities are read only as historical data stores of current data and do not need to be CORBA interfaces. Instead, it is proposed that they 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.

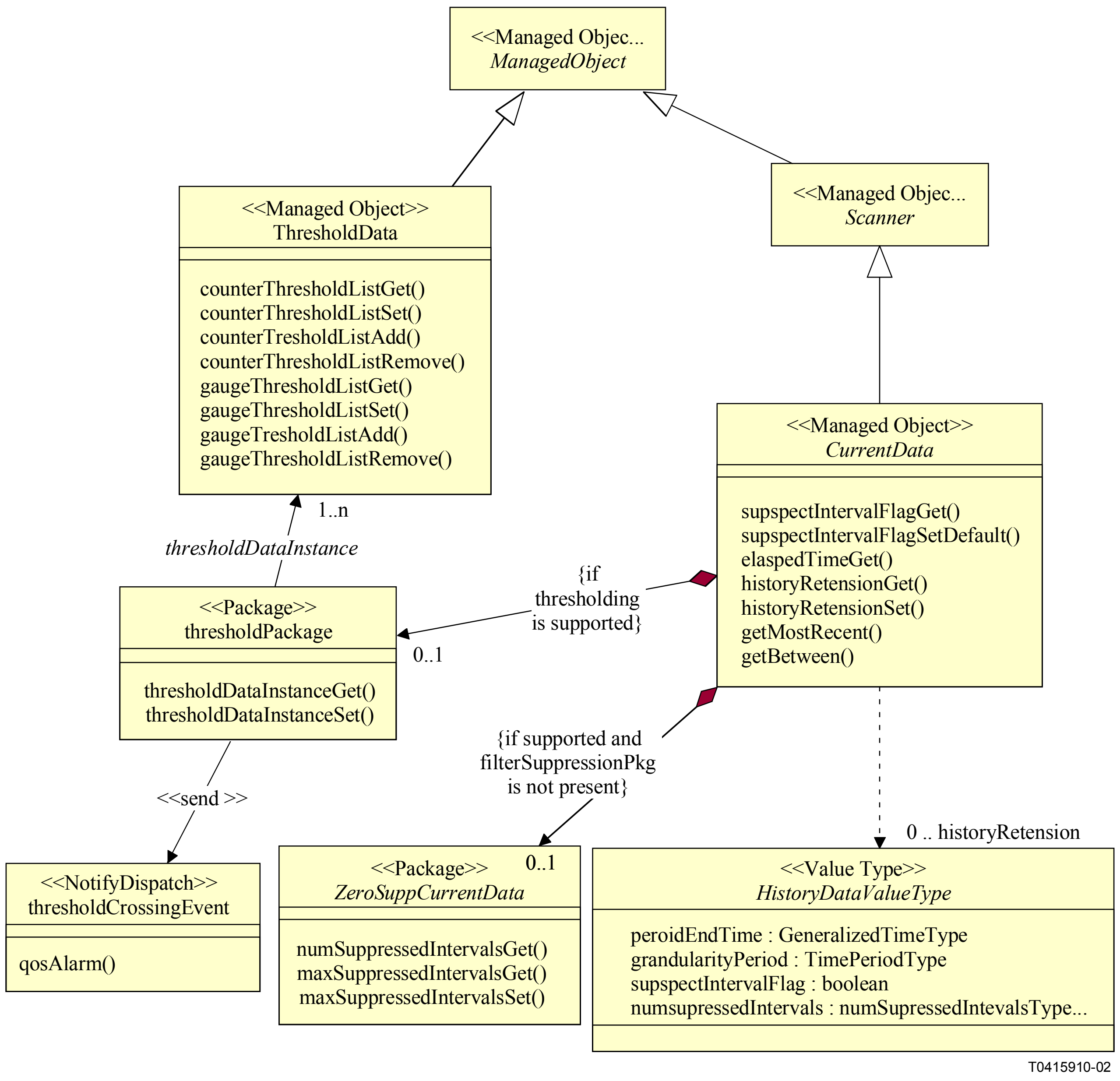
Since the data that needs to be stored is not in CORBA interfaces but HistoryDataValueTypes, to report history data, simply translating the GDMO homogeneous and simple scanners will not work. Instead, a new kind of scanner is needed – one that collects data from the HistoryDataValueTypes. This model attempts 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 recommended mechanism supports the ability to control the generation of the file, the selection of the format of the file, and file production notification to the user, along with such information as is needed to obtain the file. However, specific file transfer mechanism and file administrations are outside the scope of this Recommendation.

4.2.2 UML Model

This clause illustrates the UML model for performance management. The model covers Scanner, CurrentData, HistoryData (represented as HistoryDataValueType), ThresholdData, and HistoryDataScanner. Since the model cannot fit onto one page, it is displayed in Figures 2, 3 and 4.

The resulting model described in UML is shown in the next clause. The UML diagrams only show the model in the fine-grained approach, but these diagrams apply to both fine-grained and facade approaches. The facade IDL interface is mechanistically translated from the fine-grained IDL interface as defined in ITU-T Rec. X.780.1.



<<Managed Objec...
ManagedObject
<<Managed Object>>
ThresholdData
counterThresholdListGet()
counterThresholdListSet()
counterTresholdListAdd()
counterThresholdListRemove()
gaugeThresholdListGet()
gaugeThresholdListSet()
gaugeTresholdListAdd()
gaugeThresholdListRemove()
<<Managed Objec...
Scanner
<<Managed Object>>
CurrentData
supspectIntervalFlagGet()
supspectIntervalFlagSetDefault()
elaspedTimeGet()
historyRetensionGet()
historyRetensionSet()
getMostRecent()
getBetween()
1..n
thresholdDataInstance
<<Package>>
thresholdPackage
thresholdDataInstanceGet()
thresholdDataInstanceSet()
{if thresholding is supported}
0..1
{if supported and filterSuppressionPkg is not present}
<<send >>
<<NotifyDispatch>>
thresholdCrossingEvent
qosAlarm()
<<Package>>
ZeroSuppCurrentData
0..1
numSuppressedIntervalsGet()
maxSuppressedIntervalsGet()
maxSuppressedIntervalsSet()
0 .. historyRetension
<<Value Type>>
HistoryDataValueType
peroidEndTime : GeneralizedTimeType
grandularityPeriod : TimePeriodType
supspectIntervalFlag : boolean
numsupressedIntervals : numSupressedIntevalsType...
T0415910-02

Figure 2/Q.822.1 – Class diagram for CurrentData, HistoryData, and ThresholdData

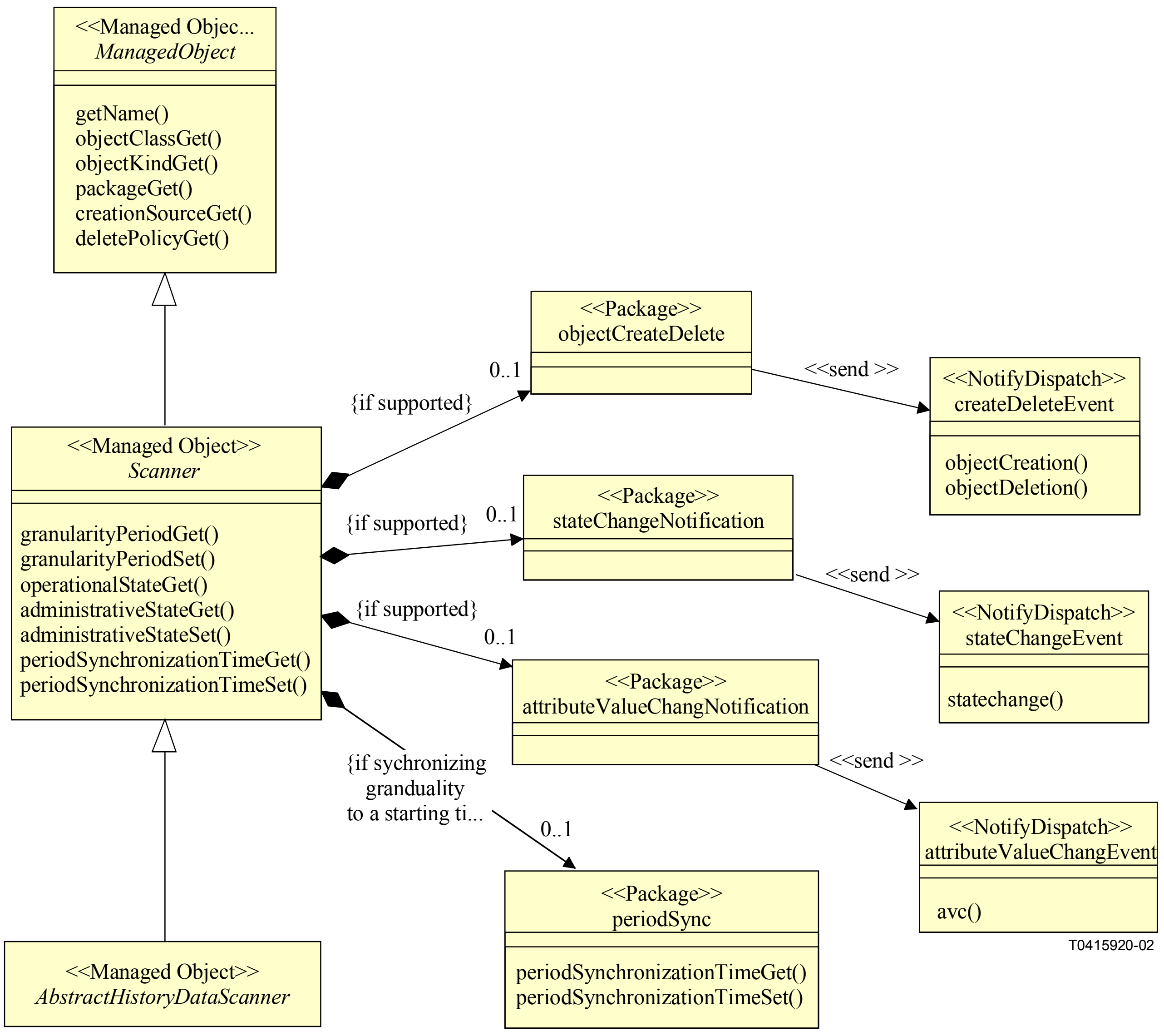


Figure 3/Q.822.1 – Class diagram for scanner

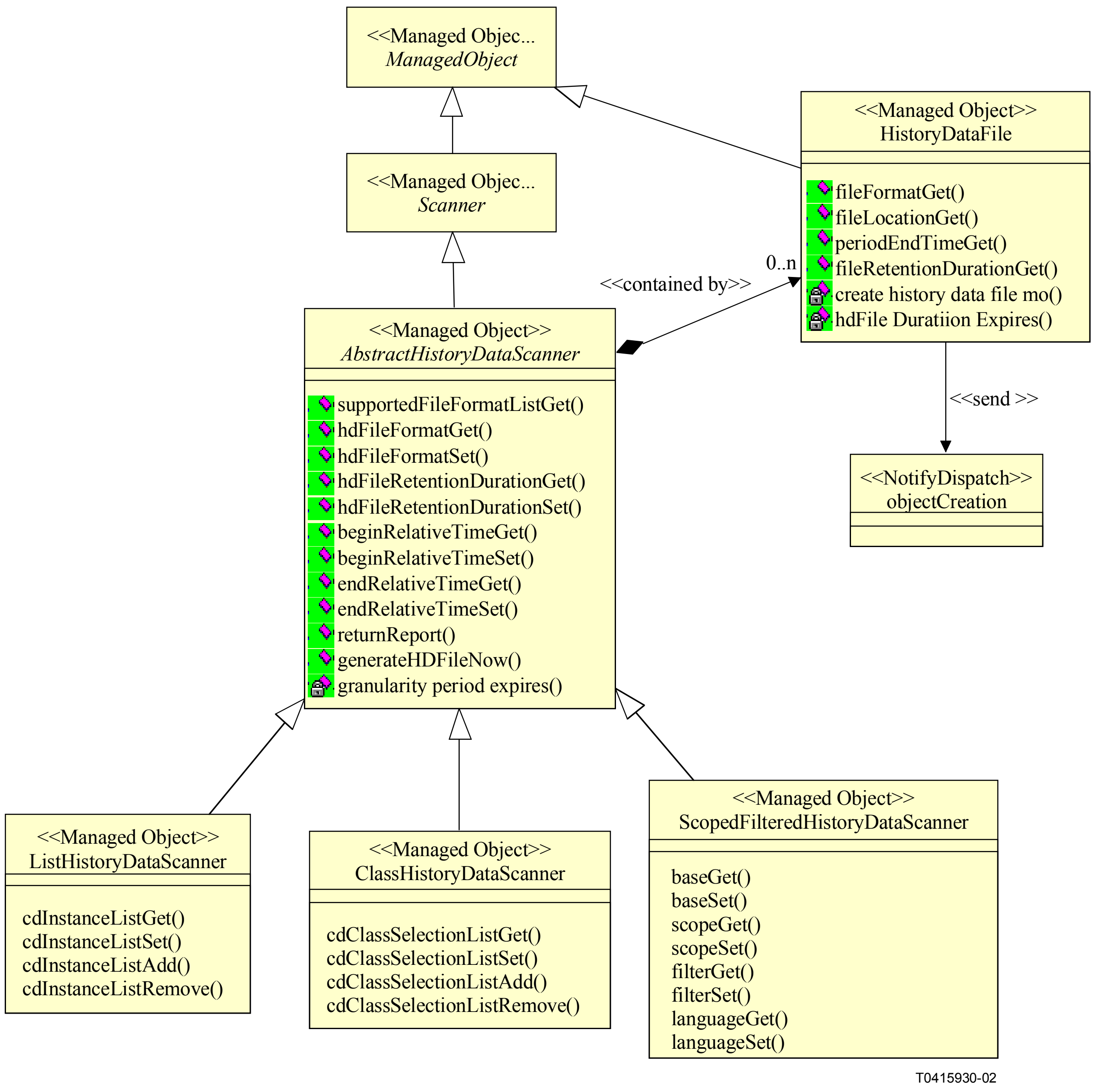
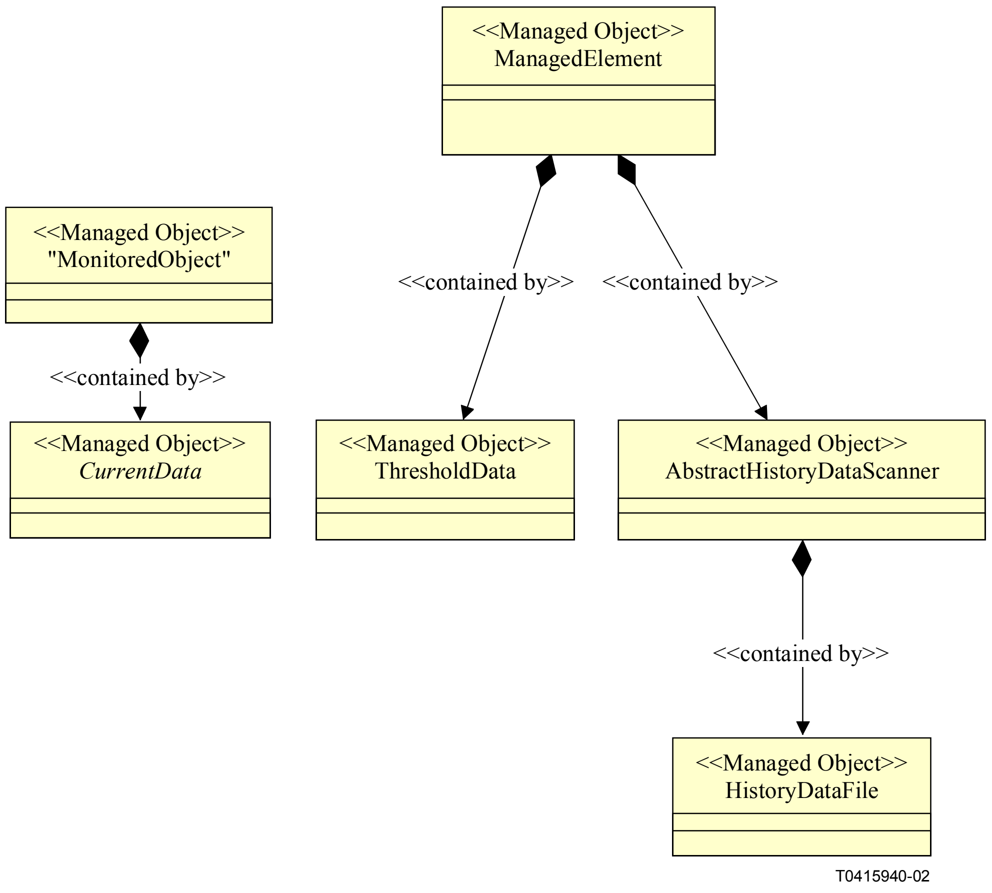


Figure 4/Q.822.1 – Class diagram for HistoryDataScanner and HistoryDataFile

4.2.3 Containment Relationship

This clause illustrates the containment relationship of the CORBA interfaces defined in this Recommendation.



<<Managed Object>> ManagedElement
<<Managed Object>> "MonitoredObject"
<<contained by>>
<<contained by>>
<<contained by>>
<<Managed Object>> *CurrentData*
<<Managed Object>> ThresholdData
<<Managed Object>> AbstractHistoryDataScanner
<<contained by>>
<<Managed Object>> HistoryDataFile
T0415940-02

Figure 5/Q.822.1 – 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 the administrative state has the value 'unlocked', the scanner is administratively permitted to perform scans. If the 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 attribute is 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 cannot 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 the 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). Furthermore, 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. 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, and false, that 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 chunks 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 occurring 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 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 notification might be emitted for a single monitored attribute, and it is possible that the notifications might be conflicting. 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 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 attribute is 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.

The zeroSuppressionPackage is a conditional package of CurrentData. When this package is present and an interval terminates with 'all-zeros' performance measurements, 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 if it reaches maxSuppressedIntervals, a history data record is created and the attribute 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 there is 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. 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 ThresholdData

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

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

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

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

If createDeleteNotificationsPackage is present, the creation or deletion of a threshold object will cause 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 attribute is defined in the counterThresholdListPackage:

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

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

The following attribute is 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

HistoryDataFile is a subclass of ManagedObject. It maps to history data file one-to-one. 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.

The managed system shall control the life-cycle of the HistoryDataFile. However, a destroy behavior that purges the history data file if HistoryDataFile object is deleted, is provided if the managing system wants to clean the history data file before expiration of file retention period.

The HistoryDataFile object has the following attributes:

- *fileLocation* – identifies file location in URL format. The time between reuse of same fileLocation for two different files shall be sufficiently beyond any possibility for confusion.
- *fileFormat* – indicates which file format used.
- *periodEndTime* – indicates the time at the end of the file generation.
- *fileRetentionDuration* – indicates file retention period before the file is purged due to garbage collection of the managed system.

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

Even though there is a HistoryDataFileFactory, a HistoryDataFile object shall never be created by the managing system. The HistoryDataFileFactory is meant to be used by the managed system.

4.3.5 AbstractHistoryDataScanner

AbstractHistoryDataScanner is a subclass of Scanner and a superclass of all other history data scanners. It is the collection of common attributes and operations of history data scanners. The AbstractHistoryDataScanner is not instantiable.

History data scanners support scanning and reporting history data by providing the ability to control the generation of the history data file, the selection of the format of the file, and the notification of user when the file is produced.

The AbstractHistoryDataScanner object has the following attributes:

- *supportedFileFormatList* – indicates the supported file formats of the managed system since it may support more than one file format.
- *hdFileFormat* – indicates which file format to use.
- *fileRetentionDuration* – indicates the file retention period for all the history data files generated by this scanner.
- *beginRelativeTime* – 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.
- *endRelativeTime* – 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 in periodical file generation until stopped by the managing system.
- *granularityPeriod (inherited from Scanner)* – sets a time window so that the history data, with time stamps that fall into this window, will be reported within a file.

Before setting any values to these attributes, the administrativeState has to be set to lock. And when the administrativeState is set to 'unlock', the periodical file generation begins. If the granularity period is zero, then there is no periodic file generation.

At the end of each granularity period of the scanner, the history data scanner scans the historical data from a set of current data objects specified in the detailed scanner, collects any data if its period end time falls within the 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 the managed system and an objectCreation notification is emitted to the managing system. The source of objectCreation notification is the HistoryDataFile object that has just been created.

The history data scanner has two operations for non-periodic history data reporting:

- *returnHDReport* – returns a sequence of history data value types which period end-time falls between now and the end of last granularity period of the scanner.
- *generateHDFileNow* – generates history data file immediately instead of waiting till the end of next granularity period of the scanner. The period end-time of history data value type in this file shall be in between now and the end of last granularity period.

4.3.6 ListHistoryDataScanner

The ListHistoryDataScanner is a specialized history data scanner and subclass of AbstractHistoryDataScanner. It returns history data reports and files based on a simple object list.

The ListHistoryDataScanner object has the following attribute:

- *cdInstancesList* – contains a set of current data objects whose historical data needs to be scanned/reported to a file.

Figure 6 diagram depicts a simple use case of ListHistoryDataScanner:

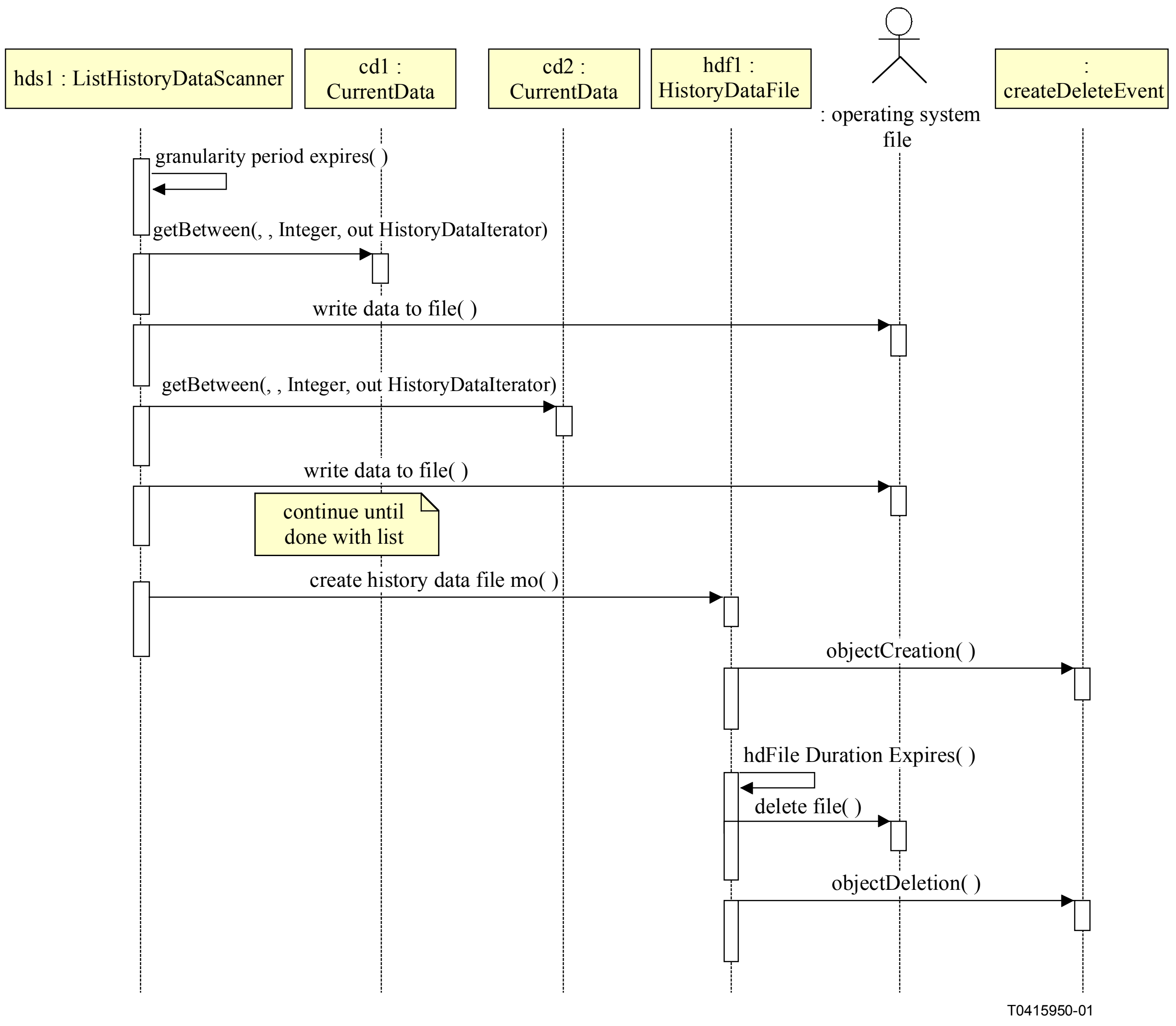

hds1 : ListHistoryDataScanner
cd1 : CurrentData
cd2 : CurrentData
hdf1 : HistoryDataFile
: operating system file
: createDeleteEvent
granularity period expires( )
getBetween(, , Integer, out HistoryDataIterator)
write data to file( )
getBetween(, , Integer, out HistoryDataIterator)
write data to file( )
continue until done with list
create history data file mo( )
objectCreation( )
hdFile Duration Expires( )
delete file( )
objectDeletion( )
T0415950-01

Figure 6/Q.822.1 – Use case of ListHistoryDataScanner

4.3.7 ClassHistoryDataScanner

The ClassHistoryDataScanner is a specialized history data scanner and subclass of AbstractHistoryDataScanner. It returns history data reports and files based on an object class list.

The ClassHistoryDataScanner object has the following attribute:

- *cdClassSelectionList* – contains a set of current data classes grouped by the containing ManagedObject (e.g. network equipment). The current data objects whose historical data needs to be scanned/reported to a file are those instances of the specified current data classes contained by the respective ManagedObject.

Before the ClassHistoryDataScanner generates a new report or file, it shall re-evaluate matching current data instances for any added or deleted current data instances since the last scanning.

4.3.8 ScopedFilteredHistoryDataScanner

The ScopedFilteredHistoryDataScanner is a specialized history data scanner and subclass of AbstractHistoryDataScanner. It returns history data reports and files based on scoping and filtering.

The ScopedFilteredHistoryDataScanner object has the following attributes:

- *base* – base-managed object used for scoping.
- *scope* – scope value as defined in ITU-T Rec. Q.816.
- *filter* – constraint filter expression that is used to evaluate the matching objects.
- *language* – a string indicating the language in which the filter expression is written.

The current data objects whose historical data needs to be scanned/reported to a file are those current data objects selected through scoping and filtering operation. Before the ScopedFilteredHistoryDataScanner generates a new report or file, it shall re-apply the scoping and filtering operation to select the latest set of current data objects.

5 Performance Management Service IDL

```
#ifndef _itut_q822_1_idl_
#define _itut_q822_1_idl_


#include <itut_x780.idl>
#include <itut_x780_1.idl>
#include <itut_x780ct.idl>
#include <itut_q816.idl>
#include <itut_q816_1.idl>
#include <itut_m3120.idl>


#pragma prefix "itu.int"


/**
This IDL code is intended to be stored in a file named "itut_q822_1.idl"
located in the search path used by IDL compilers installed on your system.
*/


/**
This module, itut_q822d1, contains IDL definition based on objects defined
in ITU-T Rec. Q.822. The IDL definitions in this file are the object interfaces.
*/
module itut_q822d1
{


/**
```

5.1 Imports

```
*/

/**
Types imported from itut_x780
*/
     typedef itut_x780::AdministrativeStateType AdministrativeStateType;
     typedef itut_x780::DeletePolicyType DeletePolicyType;
     typedef itut_x780::GeneralizedTimeType GeneralizedTimeType;
     typedef itut_x780::Istring Istring;
     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::ScopedNameSetType ScopedNameSetType;
     typedef itut_x780::UIDType UIDType;


/**
Types imported from itut_x780ct
*/
     typedef itut_x780ct::SeverityIndicatingThresholdType
          SeverityIndicatingThresholdType;
     typedef itut_x780ct::SeverityIndicatingGaugeThresholdSetType
          SeverityIndicatingGaugeThresholdSetType;
     typedef itut_x780ct::TimeIntervalType TimeIntervalType;
     typedef itut_x780ct::TimePeriodType TimePeriodType;


/**
Types imported from itut_q816
*/
     typedef itut_q816::ScopeType ScopeType;
     typedef itut_q816::FilterType FilterType;
     typedef itut_q816::LanguageType LanguageType;


/**
Types imported from itut_m3120
*/
     typedef itut_m3120::ManagedElementNameType ManagedElementNameType;
/**
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 ThresholdData;
     interface HDFile;
     interface AbstractHDScanner;
     interface ListHDScanner;
     interface ClassHDScanner;
     interface ScopedFiltedHDScanner;


/**
Valuetype forward declarations
*/
```

```
     valuetype ScannerValueType;
     valuetype CurrentDataValueType;
     valuetype HistoryDataValueType;
     valuetype ThresholdDataValueType;
     valuetype HDFileValueType;
     valuetype AbstractHDScannerValueType;
     valuetype ListHDScannerValueType;
     valuetype ClassHDScannerValueType;
     valuetype ScopedFiltedHDScannerValueType;


/**
Interface forward declarations
*/
     typedef MONameType ScannerNameType;
     typedef MONameType CurrentDataNameType;
     typedef MONameType ThresholdDataNameType;
     typedef MONameType HDFileNameType;
     typedef MONameType AbstractHDScannerNameType;
     typedef MONameType ListHDScannerNameType;
     typedef MONameType ClassHDScannerNameType;
     typedef MONameType ScopedFiltedHDScannerNameType;


/**
```

5.3 Structures and Typedefs

```
*/

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

     typedef sequence <CurrentDataNameType> CurrentDataNameSetType;

     typedef UIDType HDFileFormatType;

     typedef sequence <HDFileFormatType> HDFileFormatSetType;

     typedef Istring URLType;

     struct CounterThresholdSettingType
     {
          string                           attributeName;
          SeverityIndicatingThresholdType  threshold;
     };

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

     struct GaugeThresholdSettingType
     {
          string                      attributeName;
          SeverityIndicatingGaugeThresholdSetType threshold;
     };

/**
The order is insignificant.
*/
```

```
     typedef sequence<GaugeThresholdSettingType>
          GaugeThresholdSetType;

/**
This data structure represents all the CurrentData classes contained by a
containing object through either direct or indirect containment.
*/
     struct CDClassSelectionType
     {
          MONameType         containingObject;
          ScopedNameSetType       cdClassList;
     };

     typedef sequence<CDClassSelectionType> CDClassSelectionSetType;


/**
```

5.4 Exceptions

```
*/

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

     exception NOdeleteNotificationPackage {};

     exception NOgaugeThresholdListPackage {};

     exception NOperiodSynchronizationPackage {};

     exception NOthresholdPackage {};


     const string counterThresholdListPackage =
          "itut_q822d1::counterThresholdListPackage";
     const string deleteNotificationPackage =
          "itut_q822d1::deleteNotificationPackage";
     const string gaugeThresholdListPackage =
          "itut_q822d1::gaugeThresholdListPackage";
     const string periodSynchronizationPackage =
          "itut_q822d1::periodSynchronizationPackage";
     const string thresholdPackage =
          "itut_q822d1::thresholdPackage";


/**
Exceptions for Performance Management
*/


/**
```

5.5 Interfaces – Fine-grained

```
*/

/**
```

5.5.1 Scanner

```
See 4.3.1 for behaviour.

This interface contains the following CONDITIONAL PACKAGES.

- periodSynchronizationPackage: present 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: itut_x780::ManagedObject
     {
          AdministrativeStateType administrativeStateGet ()
               raises (itut_x780::ApplicationError);

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

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

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

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

          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 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 HistoryDataValueType

```
See 4.3.2.2 for behaviour.
*/
     valuetype HistoryDataValueType
     {
          public GeneralizedTimeType  periodEndTime;
```

```
            // GET
        public TimePeriodType       granularityPeriod;
            // GET
        public boolean              suspectIntervalFlag;
            // GET
        public short                numSupressedIntevals;
            // GET
    }; // valuetype HistoryDataValueType


/**
```

5.5.2.3 HistoryDataIterator

```
HistoryDataIterator is used for iteratively transferring a large chunk of
history data.
*/
    interface HistoryDataIterator
    {
/**
This method is used to return the next howMany history data. howMany is the
maximum number of history data for which results should be returned in the first
batch; it 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
there is no limit on the number of consecutive suppressed intervals. On the other
hand, the maxSuppressedIntervals is effectively equal to infinity.
*/
        const short maxSuppressedIntervalsDefault = -1;

        boolean suspectIntervalFlagGet ()
            raises (itut_x780::ApplicationError);
```

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

          TimeIntervalType elapsedTimeGet ()
               raises (itut_x780::ApplicationError);

          short historyRetentionGet ()
               raises (itut_x780::ApplicationError);

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

          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);

          boolean getMostRecent
               (out HistoryDataValueType historyData)
               raises (itut_x780::ApplicationError);

          HistoryDataSeqValueType getBetween
               (in GeneralizedTimeType startTime,
               in GeneralizedTimeType endTime,
               in unsigned short howMany,
               out HistoryDataIterator resultIterator)
               raises (itut_x780::ApplicationError);

/**
*/
          short numSuppressedIntervalsGet ()
               raises (itut_x780::ApplicationError);

          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 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: itut_x780::ManagedObject
     {
          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);

          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: itut_x780::ManagedObjectFactory
     {
          itut_x780::ManagedObject create
               (in NameBindingType nameBinding,
               in MONameType superior,
               in string reqID,   // auto naming if null
               out MONameType name,
               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 4.3.4 for behaviour.

This interface contains the following CONDITIONAL PACKAGES.

- objectDeleteNotificationPackage: present if an instance supports it.

```
*/
     valuetype HDFileValueType: itut_x780::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)
          CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, objectDeletion,
               deleteNotificationPackage)

     }; // interface HDFile


     interface HDFileFactory: itut_x780::ManagedObjectFactory
     {
          itut_x780::ManagedObject create
               (in NameBindingType nameBinding,
               in MONameType superior,
               in string reqID, // auto naming if empty string
               out MONameType name,
               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 4.3.5 for behaviour.
*/
     valuetype AbstractHDScannerValueType: ScannerValueType
     {
          public HDFileFormatSetType  supportedFileFormatList;
               // GET
          public HDFileFormatType     hdFileFormat;
               // GET, SET
          public TimePeriodType       hdFileRetentionDuration;
               // GET, SET
          public TimePeriodType       beginRelativeTime;
               // GET, SET
          public TimePeriodType       endRelativeTime;
               // GET, SET
     }; // valuetype HDScannerValueType


     interface AbstractHDScanner: Scanner
     {

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

/**
The hdFileFormat must be one of the supported file formats.
*/
          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);

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

          void beginRelativeTimeSet
               (in TimePeriodType beginRelativeTime)
               raises (itut_x780::ApplicationError);

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

          void endRelativeTimeSet
               (in TimePeriodType endRelativeTime)
               raises (itut_x780::ApplicationError);

          HistoryDataSeqValueType returnHDReport
               (in unsigned short howMany,
               out HistoryDataIterator resultIterator)
               raises (itut_x780::ApplicationError);

          void generateHDFileNow ()
               raises (itut_x780::ApplicationError);

     };  // interface AbstractHDScanner


/**
```

5.5.6 ListHDScanner

```
See 4.3.6 for behaviour.
*/
    valuetype ListHDScannerValueType: AbstractHDScannerValueType
    {
         public CurrentDataNameSetType    cdInstanceList;
              // GET, SET
    }; // valuetype ListHDScannerValueType


    interface ListHDScanner: AbstractHDScanner
    {
         CurrentDataNameSetType cdInstanceListGet ()
              raises (itut_x780::ApplicationError);

         void cdInstanceListSet
              (in CurrentDataNameSetType instanceList)
              raises (itut_x780::ApplicationError);

         void cdInstanceListAdd
              (in CurrentDataNameSetType instanceList)
```

```
               raises (itut_x780::ApplicationError);

          void cdInstanceListRemove
               (in CurrentDataNameSetType instanceList)
               raises (itut_x780::ApplicationError);

     }; // interface ListHDScanner


     interface ListHDScannerFactory: itut_x780::ManagedObjectFactory
     {
          itut_x780::ManagedObject create
               (in NameBindingType nameBinding,
               in MONameType superior,
               in string reqID, // auto naming if empty string
               out MONameType name,
               in CurrentDataNameSetType cdInstanceList,
               in HDFileFormatType fileFormat,
               in TimePeriodType fileRetentionDuration,
               in TimePeriodType beginRelativeTime,
               in TimePeriodType endRelativeTime,
               in AdministrativeStateType administrativeState,
               in TimePeriodType granularityPeriod,
               in GeneralizedTimeType periodSynchronizationTime)
                    // conditional
                    // periodSynchronizationPackage
               raises (itut_x780::ApplicationError,
                    itut_x780::CreateError);

     };  // interface ListHDScannerFactory


/**
```

5.5.7 ClassHDScanner

```
See 4.3.7 for behaviour.
*/
     valuetype ClassHDScannerValueType: AbstractHDScannerValueType
     {
          public CDClassSelectionSetType   cdClassSelectionList;
               // GET, SET
     }; // valuetype ClassHDScannerValueType


     interface ClassHDScanner: AbstractHDScanner
     {
          CDClassSelectionSetType cdClassSelectionListGet ()
               raises (itut_x780::ApplicationError);

          void cdClassSelectionListSet
               (in CDClassSelectionSetType classList)
               raises (itut_x780::ApplicationError);

          void cdClassSelectionListAdd
               (in CDClassSelectionSetType classList)
               raises (itut_x780::ApplicationError);

          void cdClassSelectionListRemove
               (in CDClassSelectionSetType classList)
               raises (itut_x780::ApplicationError);

     }; // interface ClassHDScanner
     interface ClassHDScannerFactory: itut_x780::ManagedObjectFactory
     {
          itut_x780::ManagedObject create
               (in NameBindingType nameBinding,
               in MONameType superior,
               in string reqID, // auto naming if empty string
               out MONameType name,
               in CDClassSelectionSetType cdClassSelectionList,
               in HDFileFormatType fileFormat,
               in TimePeriodType fileRetentionDuration,
               in TimePeriodType beginRelativeTime,
               in TimePeriodType endRelativeTime,
               in AdministrativeStateType administrativeState,
               in TimePeriodType granularityPeriod,
               in GeneralizedTimeType periodSynchronizationTime)
                    // conditional
                    // periodSynchronizationPackage
               raises (itut_x780::ApplicationError,
                    itut_x780::CreateError);

     };  // interface ClassHDScannerFactory


/**
```

5.5.8 ScopedFilteredHDScanner

```
See 4.3.8 for behaviour.
*/
     valuetype ScopedFiltedHDScannerValueType: AbstractHDScannerValueType
     {
          public MONameType       base;
               // GET, SET
          public ScopeType        scope;
               // GET, SET
          public FilterType       filter;
               // GET, SET
          public LanguageType     language;
               // GET, SET
     }; // valuetype ScopedFiltedHDScannerValueType


     interface ScopedFiltedHDScanner: AbstractHDScanner
     {
          MONameType baseGet ()
               raises (itut_x780::ApplicationError);

          void baseSet
               (in MONameType base)
               raises (itut_x780::ApplicationError);

          ScopeType scopeGet ()
               raises (itut_x780::ApplicationError);

          void scopeSet
               (in ScopeType scope)
               raises (itut_x780::ApplicationError);

          FilterType filterGet ()
               raises (itut_x780::ApplicationError);

          void filterSet
               (in FilterType filter)
```

```
                raises (itut_x780::ApplicationError);

           LanguageType languageGet ()
                raises (itut_x780::ApplicationError);

           void languageSet
                (in LanguageType language)
                raises (itut_x780::ApplicationError);

     }; // interface ScopedFiltedHDScanner


     interface ScopedFiltedHDScannerFactory:
          itut_x780::ManagedObjectFactory
     {
          itut_x780::ManagedObject create
                (in NameBindingType nameBinding,
                in MONameType superior,
                in string reqID, // auto naming if empty string
                out MONameType name,
                in MONameType base,
                in ScopeType scope,
                in FilterType filter,
                in LanguageType language,
                in HDFileFormatType fileFormat,
                in TimePeriodType fileRetentionDuration,
                in TimePeriodType beginRelativeTime,
                in TimePeriodType endRelativeTime,
                in AdministrativeStateType administrativeState,
                in TimePeriodType granularityPeriod,
                in GeneralizedTimeType periodSynchronizationTime)
                     // conditional
                     // periodSynchronizationPackage
                raises (itut_x780::ApplicationError,
                     itut_x780::CreateError);

     };  // interface ScopedFiltedHDScannerFactory


/**
```

5.6 Interfaces – Facade

```
The facade history data scanners are defined here as per ITU-T Rec. X.780.1.
However the fine-grained history data scanners would be sufficient for facade
implementation.
*/

/**
```

5.6.1 Scanner_F

```
See 4.3.1 for behaviour.

This interface contains the following CONDITIONAL PACKAGES.

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

The Scanner interface is not instantiable.
*/
     interface Scanner_F: itut_x780::ManagedObject_F
     {
          AdministrativeStateType administrativeStateGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

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

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

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

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

          GeneralizedTimeType periodSynchronizationTimeGet
               (in MONameType name)
               raises (itut_x780::ApplicationError,
                    NOperiodSynchronizationPackage);

          void periodSynchronizationTimeSet
               (in MONameType name,
               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_F


/**
```

5.6.2 CurrentData_F

```
See 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.
*/
```

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

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

          boolean suspectIntervalFlagGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

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

          TimeIntervalType elapsedTimeGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          short historyRetentionGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

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

          MONameSetType thresholdDataInstanceListGet
               (in MONameType name)
               raises (itut_x780::ApplicationError,
                    NOthresholdPackage);

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

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

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

          boolean getMostRecent
               (in MONameType name,
               out HistoryDataValueType historyData)
               raises (itut_x780::ApplicationError);
          HistoryDataSeqValueType getBetween
               (in MONameType name,
               in GeneralizedTimeType startTime,
               in GeneralizedTimeType endTime,
               in unsigned short howMany,
               out HistoryDataIterator resultIterator)
               raises (itut_x780::ApplicationError);

          short numSuppressedIntervalsGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          short maxSuppressedIntervalsGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

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

          CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, qualityOfServiceAlarm,
               NOthresholdPackage)

     };  // interface CurrentData_F


/**
```

5.6.3 ThresholdData_F

```
See 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.
*/
     interface ThresholdData_F: itut_x780::ManagedObject_F
     {
          CounterThresholdSetType counterThresholdListGet
               (in MONameType name)
               raises (itut_x780::ApplicationError,
                    NOcounterThresholdListPackage);

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

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

          GaugeThresholdSetType gaugeThresholdListGet
               (in MONameType name)
               raises (itut_x780::ApplicationError,
                    NOgaugeThresholdListPackage);

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

          void gaugeThresholdListAdd
               (in MONameType name,
               in GaugeThresholdSetType gaugeThresholdList)
               raises (itut_x780::ApplicationError,
                    NOgaugeThresholdListPackage);

          void gaugeThresholdListRemove
               (in MONameType name,
               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_F


/**
```

5.6.4 HDFile_F (HistoryDataFile)

```
See 4.3.4 for behaviour.

This interface contains the following CONDITIONAL PACKAGES.

- objectDeleteNotificationPackage: present if an instance supports it.
*/
     interface HDFile_F: itut_x780::ManagedObject_F
     {
          HDFileFormatType fileFormatGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          URLType fileLocationGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);
          GeneralizedTimeType periodEndTimeGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

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

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

     }; // interface HDFile_F


/**
```

5.6.5 AbstractHDScanner_F (AbstractHistoryDataScanner)

```
See 4.3.5 for behaviour.
*/
     interface AbstractHDScanner_F: Scanner_F
     {

          HDFileFormatSetType supportedFileFormatListGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

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

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

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

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

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

          void beginRelativeTimeSet
               (in MONameType name,
               in TimePeriodType beginRelativeTime)
               raises (itut_x780::ApplicationError);

          TimePeriodType endRelativeTimeGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);
          void endRelativeTimeSet
               (in MONameType name,
               in TimePeriodType endRelativeTime)
               raises (itut_x780::ApplicationError);

          HistoryDataSeqValueType returnHDReport
               (in MONameType name,
               in unsigned short howMany,
               out HistoryDataIterator resultIterator)
               raises (itut_x780::ApplicationError);

          void generateHDFileNow
               (in MONameType name)
               raises (itut_x780::ApplicationError);

     };  // interface AbstractHDScanner_F


/**
```

5.6.6 ListHDScanner_F

```
See 4.3.6 for behaviour.
*/
     interface ListHDScanner_F: AbstractHDScanner_F
     {
          CurrentDataNameSetType cdInstanceListGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void cdInstanceListSet
               (in MONameType name,
               in CurrentDataNameSetType instanceList)
               raises (itut_x780::ApplicationError);

          void cdInstanceListAdd
               (in MONameType name,
               in CurrentDataNameSetType instanceList)
               raises (itut_x780::ApplicationError);

          void cdInstanceListRemove
               (in MONameType name,
               in CurrentDataNameSetType instanceList)
               raises (itut_x780::ApplicationError);

     }; // interface ListHDScanner_F


/**
```

5.6.7 ClassHDScanner_F

```
See 4.3.7 for behaviour.
*/
     interface ClassHDScanner_F: AbstractHDScanner_F
     {
          CDClassSelectionSetType cdClassSelectionListGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void cdClassSelectionListSet
               (in MONameType name,
               in CDClassSelectionSetType classList)
               raises (itut_x780::ApplicationError);
          void cdClassSelectionListAdd
               (in MONameType name,
               in CDClassSelectionSetType classList)
               raises (itut_x780::ApplicationError);

          void cdClassSelectionListRemove
               (in MONameType name,
               in CDClassSelectionSetType classList)
               raises (itut_x780::ApplicationError);

     }; // interface ClassHDScanner_F


/**
```

5.6.8 ScopedFilteredHDScanner_F

```
See 4.3.8 for behaviour.
*/
     interface ScopedFiltedHDScanner_F: AbstractHDScanner_F
     {
          MONameType baseGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void baseSet
               (in MONameType name,
               in MONameType base)
               raises (itut_x780::ApplicationError);

          ScopeType scopeGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void scopeSet
               (in MONameType name,
               in ScopeType scope)
               raises (itut_x780::ApplicationError);

          FilterType filterGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void filterSet
               (in MONameType name,
               in FilterType filter)
               raises (itut_x780::ApplicationError);

          LanguageType languageGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void languageSet
               (in MONameType name,
               in LanguageType language)
               raises (itut_x780::ApplicationError);

     }; // interface ScopedFiltedHDScanner_F


/**
```

5.7 Notifications

```
No model specific notifications defined in this Recommendation.
*/


/**
```

5.8 Name Binding

```
*/

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


/**
```

5.8.1 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_q822d1::ThresholdData";
               const boolean subordinateSubclassesAllowed = TRUE;
               const boolean managerCreatesAllowed = TRUE;
               const DeletePolicyType deletePolicy =
                    itut_x780::deleteOnlyIfNoContainedObjects;
               const string  kind = "ThresholdData";
          }; // module ThresholdData_ManagedElement


/**
```

5.8.2 HDFile

```
This name binding is used to name the HDFile object to an AbstractHDScanner
object.
*/
          module HDFile_AbstractHDScanner
          {
               const string  superiorClass =
                    "itut_q822d1::AbstractHDScanner";
               const boolean superiorSubclassesAllowed = TRUE;
               const string  subordinateClass =
                    "itut_q822d1::HDFile";
               const boolean subordinateSubclassesAllowed = TRUE;
               const boolean managerCreatesAllowed = FALSE;
               const DeletePolicyType deletePolicy =
                    itut_x780::deleteOnlyIfNoContainedObjects;
               const string  kind = "HDFile";
          }; // module HDFile_AbstractHDScanner


/**
```

5.8.3 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_q822d1::AbstractHDScanner";
               const boolean subordinateSubclassesAllowed = TRUE;
               const boolean managerCreatesAllowed = TRUE;
               const DeletePolicyType deletePolicy =
                    itut_x780::deleteOnlyIfNoContainedObjects;
               const string  kind = "HDScanner";
          }; // module AbstractHDScanner_ManagedElement


     }; // module NameBinding


/**
```

5.9 HDFileFormatConst

This module contains constant values identifying history data file formats.

```
*/
     module HDFileFormatConst
     {
          const string moduleName =
               "itut_q822d1::HDFileFormatConst";

          const short unknown = 0;

          const short q822d1FileText = 1;

          const short q822d1FileXML = 2;

     }; // HDFileFormatConst


}; // module itut_q822d1




#endif // _itut_q822_1_idl_
```

ANNEX A

File Format

This annex describes the format of a performance measurement data file, named q822d1FileText.

A.1 Text File Format BNF

The BNF in this annex defines the format of performance measurement data file, named "q822d1FileText".

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

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

<FileFormat> := <String>

<FormatVersion> := <String>

<ModelParadigm> := <String>

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

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

<NodeID> := <String>

<MeasuredObjectIDPrefix> := <ID>
     | <Empty>

<Tables> := <MeasuredObjectIDAliases> <Table> <Tables>
     | <MeasuredObjectIDAliases> <Table>

<MeasuredObjectIDAliases> := <MeasuredObjectIDAlias> <MeasuredObjectIDAliases>
     | <MeasuredObjectIDAlias>
     | <Empty>

<MeasuredObjectIDAlias> := #idalias <FD> <ShortID> <FD>     <LongID> <Newline>

<ShortID> := <ID>

<LongID> := <ID>

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

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

<MeasuredObjectType> := <String>

<GranularityPeriod> := <TimePeriodValue>

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

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

<DataSetTypeName> := <String>
```

```
<DataSetTypeIndex> := <IntegerString>

<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> := <MeasuredObjectID> <FD> <DataSetValues> <Newline>

<MeasuredObjectID> := <ID>  // can be aliased and prefixed

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

<DataSetValue> := <DataSetTypeIndex> <FD> <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

<ID> := <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) characters}

<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
```

NOTE 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.

NOTE 2 – A backward compatible file parser shall skip and ignore any data between unknown #<tag> and #end<tag>.

NOTE 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 Figure 7 which depicts the file structure):

1) The file type is defined by its file format, format version, and modelling paradigm. FileFormat indicates which file format this file follows. FormatVersion specifies a version of particular file format. The value for FileFormat of this Recommendation is "q822d1FileText", and FormatVersion is "version1".
2) The modelling paradigm is the modelling paradigm of management system that generates the file. If the file is generated from a CORBA management system compliant with ITU-T CORBA framework, the value for ModelParadigm shall be "ITUTCORBA". Otherwise, an appropriate value shall be provided, for instance, "CMIP", "SNMP", "TL1", "abcCORBA", etc.
3) The performance management specification for a specific information modelling domain shall specify the semantics and naming convention of ModelParadigm, NodeID, MeasuredObjectType, DataSetTypeName, ParameterType, ID, MeasuredObjectIDPrefix, MeasuredObjectID.
4) MeasuredObjectIDPrefix defines the common prefix for MeasuredObjectID in a node section.
5) MeasuredObjectIDAlias defines the short alias MeasuredObjectID for long FDN MeasuredObjectID. The scope of an alias definition extends to end of file unless redefined by another MeasuredObjectIDAlias.

6) DataSetTypeIndex of a DataSetValue indicates the corresponding data set type as defined in the table header.

7) All the measurement data within a table have the same granularity period and represent measurements for the same measured object type.

8) If there is no data for a particular period, the file shall still contain #period and #endperiod with empty information for that period.

```
#file:file format:format version:modelling paradigm
    #node:node id:measured object id prefix          // first node, measured object id prefix is optional
        #idalias:short id:long id                    // measured object id aliases are optional
        …                                            // measured object id aliases continued
        #table                             // first table
                #header:measured object type:granularity period
                        #dataset:data set type name:data set type index
                                                              // first date set type
                                suspect:parameter type:parameter type:…:
                        #enddataset
                        #dataset:data set type name:data set type index
                                                              // second data set type
                                suspect:parameter type:parameter type:…:
                        #enddataset
                …                                             // data set type continued
                #endheader
                #period:period end time        // first period values
                        measured object id:    // first measured object
                                data set type index:suspect flag:value:value:…:
                                                              // first date set values
                                data set type index:suspect flag:value:value:…:
                                                              // second data set values
                                …:                            // data set values continued
                        measured object id:    // second measured object
                                data set type index:suspect flag:value:value:…:
                                                              // first date set values
                                data set type index:suspect flag:value:value:…:
                                                              // second data set values
                                …:                            // data set values continued
                        …                      // measured object continued
                #endperiod
                #period:period end time        // second period values
                        …                      // measured objects
                #endperiod
                …                              // period values continued
        #endtable
        #idalias:short id:long id                    // measured object id aliases are optional
        …                                            // measured object id aliases continued
        #table                             // second table
        …
        #endtable
        …                                  // table continued
    #endnode
    #node:node id:measured object id prefix          // second node, measured object id prefix is optional
    …
    #endnode
    …                                      // node continued
#endfile
```

Figure 7/Q.822.1 – Illustration of PM File Format

ANNEX B

CORBA File Format Convention

This annex describes the semantics and naming convention of ModelParadigm, NodeID, MeasuredObjectType, DataSetTypeName, ParameterType, ID, MeasuredObjectIDPrefix, MeasuredObjectName in ITU-T CORBA framework.

1) *ModelParadigm* – shall be "ITUTCORBA".

2) *NodeID* – FDN of managed element, or application specific name of managed element.

3) *MeasuredObjectType* – Scoped interface name of measured managed object.

4) *DataSetTypeName* – Scoped interface name of CurrentData or its subclass.

5) *ParameterType* – Name of attribute in HistoryDataValueType or its sub-valuetype.

6) *ID* – FDN of CORBA name of managed object.

7) *MeasuredObjectIDPrefix* – RDN of CORBA name of managed object.

8) *MearsuredObjectID* – FDN of CORBA name of measured managed object.

APPENDIX I

Example of CurrentDataValueType and HistoryDataValueType inheritance usage

As described in 4.3.2, 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 appendix provides an example of 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: itut_q822d1::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: itut_q822d1::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: itut_q822d1::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 through the TrafficControlCurrentDataFactory interface:

```
interface TrafficControlCurrentDataFactory: itut_x780::ManagedObjectFactory
{
     itut_x780::ManagedObject create
               (in NameBindingType nameBinding,
               in MONameType superior,
               in string reqID,   // auto naming if null
               out MONameType name,
               in IstringSetType packageNameList,
               in short historyRetention,
                    // Mandatory now (used to be conditional)
                    // GET-REPLACE
               in AdministrativeStateType administrativeState,
                    // GET-REPLACE
               in TimePeriodType granularityPeriod,
                    // GET-REPLACE
               in MONameSetType thresholdDataInstanceList,
                    // conditional
                    // thresholdPackage
                    // GET-REPLACE ADD-REMOVE
               in short maxSuppressedIntervals,
                    // conditional
                    // zeroSuppressionPackage
                    // GET-REPLACE
               in GeneralizedTimeType periodSynchronizationTime)
                    // conditional
                    // periodSynchronizationPackage
                    // GET-REPLACE
               raises (itut_x780::ApplicationError,
                    itut_x780::CreateError);

}; // interface TrafficControlCurrentDataFactory
```

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 two methods: getMostRecent() and getBetween() from the instance of TrafficControlCurrentData.

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

Printed in Switzerland
Geneva, 2002