UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T Q.822.1

(10/2001)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

SERIE Q: CONMUTACIÓN Y SEÑALIZACIÓN

Interfaz Q3

Servicio de gestión de la calidad de funcionamiento de la RGT basado en arquitectura de intermediario de petición de objeto común

Recomendación UIT-T Q.822.1

RECOMENDACIONES UIT-T DE LA SERIE Q

CONMUTACIÓN Y SEÑALIZACIÓN

SEÑALIZACIÓN EN EL SERVICIO MANUAL INTERNACIONAL	Q.1–Q.3
EXPLOTACIÓN INTERNACIONAL SEMIAUTOMÁTICA Y AUTOMÁTICA	Q.4–Q.59
FUNCIONES Y FLUJOS DE INFORMACIÓN PARA SERVICIOS DE LA RDSI	Q.60–Q.99
CLÁUSULAS APLICABLES A TODOS LOS SISTEMAS NORMALIZADOS DEL UIT-T	Q.100–Q.119
ESPECIFICACIONES DE LOS SISTEMAS DE SEÑALIZACIÓN N.° 4 Y N.° 5	Q.120–Q.249
ESPECIFICACIONES DEL SISTEMA DE SEÑALIZACIÓN N.° 6	Q.250–Q.309
ESPECIFICACIONES DEL SISTEMA DE SEÑALIZACIÓN R1	Q.310–Q.399
ESPECIFICACIONES DEL SISTEMA DE SEÑALIZACIÓN R2	Q.400–Q.499
CENTRALES DIGITALES	Q.500–Q.599
INTERFUNCIONAMIENTO DE LOS SISTEMAS DE SEÑALIZACIÓN	Q.600–Q.699
ESPECIFICACIONES DEL SISTEMA DE SEÑALIZACIÓN N.° 7	Q.700–Q.799
Generalidades	Q.700
Parte transferencia de mensajes	Q.701–Q.709
Parte control de la conexión de señalización	Q.711–Q.719
Parte usuario de telefonía	Q.720–Q.729
Servicios suplementarios de la RDSI	Q.730–Q.739
Parte usuario de datos	Q.740–Q.749
Gestión del sistema de señalización N.° 7	Q.750–Q.759
Parte usuario de la RDSI	Q.760–Q.769
Parte aplicación de capacidades de transacción	Q.770–Q.779
Especificaciones de las pruebas	Q.780–Q.799
INTERFAZ Q3	**Q.800–Q.849**
SISTEMA DE SEÑALIZACIÓN DIGITAL DE ABONADO N.° 1	Q.850–Q.999
Generalidades	Q.850–Q.919
Capa de enlace de datos	Q.920–Q.929
Capa de red	Q.930–Q.939
Gestión usuario-red	Q.940–Q.949
Descripción de la etapa 3 para los servicios suplementarios que utilizan el sistema de señalización digital de abonado N.° 1	Q.950–Q.999
RED MÓVIL TERRESTRE PÚBLICA	Q.1000–Q.1099
INTERFUNCIONAMIENTO CON SISTEMAS MÓVILES POR SATÉLITE	Q.1100–Q.1199
RED INTELIGENTE	Q.1200–Q.1699
REQUISITOS Y PROTOCOLOS DE SEÑALIZACIÓN PARA IMT-2000	Q.1700–Q.1799
ESPECIFICACIONES DE LA SEÑALIZACIÓN RELACIONADA CON EL CONTROL DE LLAMADA INDEPENDIENTE DEL PORTADOR	Q.1900–Q.1999
RED DIGITAL DE SERVICIOS INTEGRADOS DE BANDA ANCHA (RDSI-BA)	Q.2000–Q.2999

Para más información, véase la Lista de Recomendaciones del UIT-T.

Recomendación UIT-T Q.822.1

Servicio de gestión de la calidad de funcionamiento de la RGT basado en arquitectura de intermediario de petición de objeto común

Resumen

Esta Recomendación define un modelo de información para uso en la gestión de la calidad de funcionamiento de las telecomunicaciones, basado en CORBA. Define, en lenguaje de definición de interfaz (IDL), un conjunto de interfaces, notificaciones y constantes. El propósito de esta Recomendación es definir un modelo CORBA/IDL similar al definido en las Recomendaciones UIT-T X.739 y Q.822 utilizando CMISE. La presente Recomendación cumple con las normas de modelado CORBA de las Recomendaciones UIT-T X.780, X.780.1, Q.816, Q.816.1 y M.3120.

Orígenes

La Recomendación UIT-T Q.822.1, preparada por la Comisión de Estudio 4 (2001-2004) del UIT-T, fue aprobada por el procedimiento de la Resolución 1 de la AMNT el 7 de octubre de 2001.

PREFACIO

La UIT (Unión Internacional de Telecomunicaciones) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones. El UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Asamblea Mundial de Normalización de las Telecomunicaciones (AMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución 1 de la AMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT no ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB.

ÍNDICE

Recomendación UIT-T Q.822.1

Servicio de gestión de la calidad de funcionamiento de la RGT basado en arquitectura de intermediario de petición de objeto común

1 Alcance

Esta Recomendación define un servicio de soporte para uso en la gestión de la calidad de funcionamiento de las telecomunicaciones basado en CORBA. Define un conjunto de interfaces (en la versión granularidad fina y en la versión fachada), tipos de datos y constantes utilizando para ello el lenguaje de definición de interfaz (IDL, *interface definition language*). El propósito de esta Recomendación es definir una especificación CORBA/IDL similar a la definida en las Recomendaciones UIT-T X.739 y Q.822 utilizando CMISE. La presente Recomendación cumple con las normas de modelado CORBA de las Recomendaciones UIT-T X.780, X.780.1, Q.816, Q.816.1 y M.3120.

2 Referencias

Las siguientes Recomendaciones del UIT-T y otras referencias contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y otras referencias son objeto de revisiones por lo que se preconiza que los usuarios de esta Recomendación investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y otras referencias citadas a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T actualmente vigentes.

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

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

[3] Recomendación UIT-T Q.816 (2001), *Servicios de la RGT basados en la arquitectura de intermediario de petición de objeto común*.

[4] Recomendación UIT-T Q.816.1 (2001), *Extensiones a los servicios de la RGT basados en la arquitectura de intermediario de petición de objeto común para el soporte de interfaces de granularidad gruesa*.

[5] Recomendación UIT-T X.780 (2001), *Directrices de la RGT para definir objetos gestionados de la arquitectura de intermediario de petición de objeto común*.

[6] Recomendación UIT-T X.780.1 (2001), *Directrices de la RGT para definir objetos gestionados de la arquitectura de intermediario de petición de objeto común de granularidad gruesa*.

[7] Recomendación UIT-T M.3120 (2001), *Modelo de información de red genérico basado en la arquitectura de intermediario de petición de objeto común y a nivel de elemento de red*.

[8] Recomendación UIT-T X.721 (1992), *Tecnología de la información – Interconexión de sistemas abiertos – Estructura de la información de gestión: Definición de la información de gestión*.

[9] Recomendación UIT-T X.738 (1993), *Tecnología de la información – Interconexión de sistemas abiertos – Gestión de sistemas: Función de sumario*.

[10] Recomendación UIT-T X.739 (1993), *Tecnología de la información – Interconexión de sistemas abiertos – Gestión de sistemas: Objetos métricos y atributos*.

[11] Recomendación UIT-T Q.822 (1994), *Descripción de la etapa 1, de la etapa 2 y de la etapa 3 para la interfaz Q3 – Gestión de la calidad de funcionamiento*.

[12] Recomendación UIT-T Q.823 (1996), *Especificaciones funcionales de las etapas 2 y 3 para la gestión del tráfico*.

3 Abreviaturas

En esta Recomendación se utilizan las siguientes siglas.

CMIP Protocolo común de información de gestión (*common management information protocol*)

CORBA Arquitectura de intermediario de petición de objeto común (*common object request broker architecture*)

DN Nombre distinguido (*distinguished name*)

FDN Nombre distinguido completo (*fully distinguished name*)

GDMO Directrices para la definición de objetos gestionados (*guidelines for the definition of managed objects*)

NE Elemento de red (*network element*)

PM Gestión de la calidad de funcionamiento (*performance management*)

QoS Calidad del servicio (*quality of service*)

RDN Nombre distinguido relativo (*relative distinguished name*)

RGT Red de gestión de las telecomunicaciones

SNMP Protocolo simple de gestión de red (*simple network management protocol*)

TL1 Idioma de transacción 1 (*transaction language 1*)

UML Lenguaje de modelado unificado (*unified modelling language*)

4 Visión general del servicio de gestión de la calidad de funcionamiento

4.1 Requisitos

4.1.1 Requisitos funcionales

La medición de la calidad de funcionamiento se utiliza de dos formas en la RGT. Una es la medición de la calidad de funcionamiento de entidades de transporte y protocolo. En este caso las mediciones se someten a una comparación con un valor umbral y sus resultados se recopilan para formar un historial técnico y de servicio, no necesariamente en tiempo real. Ésta es la utilización soportada por la Rec. UIT-T Q.822 y las normas específicas de tecnología basadas directamente en esta Recomendación. La otra forma de utilizar la medición de la calidad de funcionamiento es en soporte de la gestión del tráfico de red. En esta aplicación, las mediciones no se comparan con un valor umbral sino que sus resultados se recogen periódicamente en una escala de tiempo necesaria para aplicar controles de gestión de red. La escala de tiempo típica que normalmente se ha venido utilizando es de 5 minutos. Ésta es la utilización soportada por la Rec. UIT-T Q.823. Dicha Recomendación reutiliza el mecanismo de la Rec. UIT-T Q.822 junto con el explorador simple genérico de la Rec. UIT-T X.738 para producir un informe de exploración individual que sumariza el periodo de granularidad precedente y se informa al sistema de operaciones de la gestión de red. Estas

dos formas de utilizar la medición de la calidad de funcionamiento se resumen en los casos de utilización representados en la figura 1.

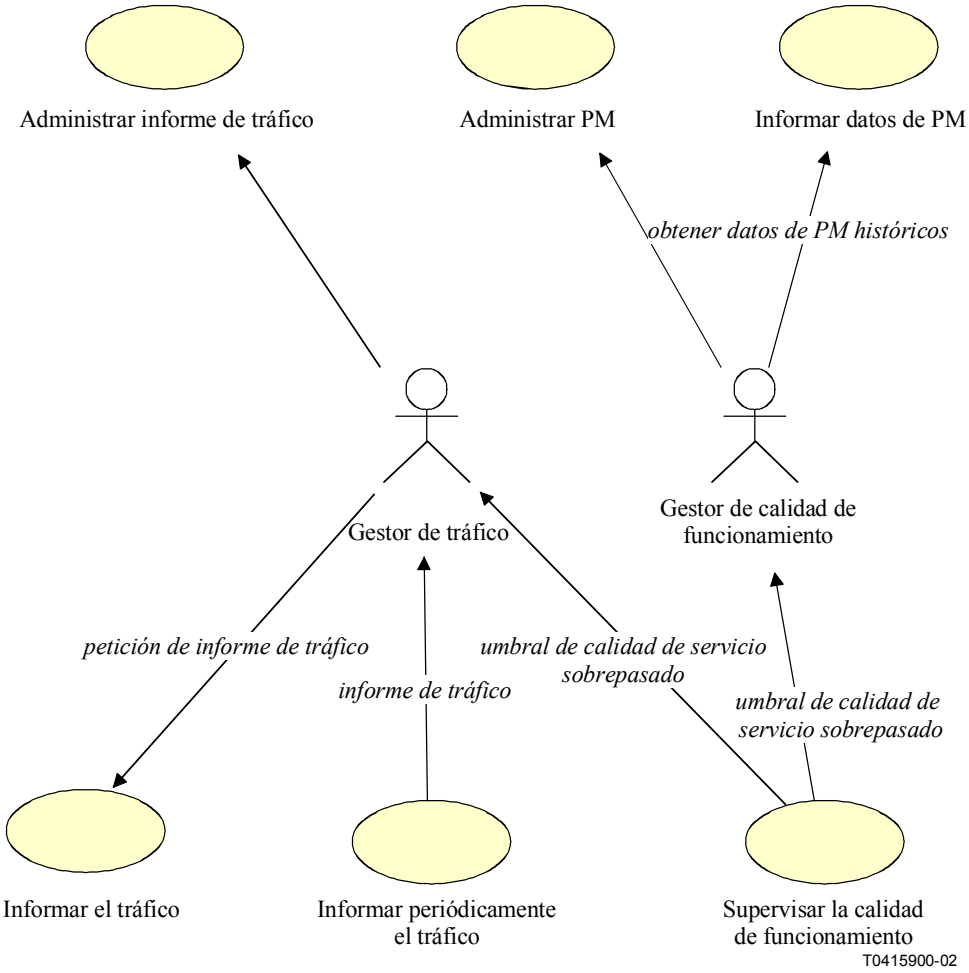

Figura 1/Q.822.1 – Casos de utilización de la medición de la calidad de funcionamiento

En resumen, entre las funciones de gestión de la RGT para la gestión de la calidad de funcionamiento está la recopilación de datos, el almacenamiento de datos, la comparación con valores de umbral y los informes de datos.

Por recopilación de datos de calidad de funcionamiento se entiende la aptitud de un elemento de red para recopilar los diversos datos de PM relacionados con una entidad supervisada individual en ese elemento de red. Esta función permite al encargado de la RGT asignar intervalos de recopilación de datos de PM, suspender o reanudar el proceso de recopilación de datos de PM, y reiniciar los contadores de supervisión de la calidad de funcionamiento.

Por almacenamiento de los datos PM se entiende la aptitud de un elemento de red para almacenar datos PM históricos sobre cada entidad supervisada durante un tiempo determinado. Esta función permite al encargado de la RGT especificar el tiempo durante el cual se realizará un registro específico de los datos PM históricos, para clasificar los datos históricos sobre la base de cierto criterio (por ejemplo, eliminar los datos "todos ceros"), y suprimir los datos PM históricos al final del intervalo de tiempo.

Por comparación de valores umbral PM se entiende la aptitud de un elemento de red para informar al encargado de la RGT que se ha rebasado un umbral. Además, permite al encargado de la RGT establecer criterios de comparación de umbrales.

Por informes de datos PM se entiende la aptitud de un elemento de red para señalar datos PM regularmente o a raíz de una petición espontánea del encargado de la RGT. Los informes pueden contener datos sobre una entidad supervisada dada o datos sumarizados de un conjunto de entidades supervisadas. Esta función permite al encargado de la RGT solicitar datos PM, permitir/prohibir la emisión de informes regulares en el elemento de red, y clasificar los informes de datos PM en base a algún criterio (por ejemplo, supresión de los datos "todos ceros").

4.1.2 Requisitos de diseño

Esta Recomendación sigue las directrices de modelado de información definidas en la Rec. UIT-T X.780. Además con esta Recomendación se pretende que el modelo sea práctico, sencillo y tenga sentido para los usuarios y fabricantes.

4.2 Modelado del servicio

4.2.1 Alcance

Para el soporte de los requisitos de funcionamiento, el servicio de gestión de la calidad de funcionamiento necesita currentData para recopilar los resultados de las mediciones de calidad de funcionamiento, historyData para almacenar dichos resultados de mediciones recopilados, thresholdData para especificar los umbrales y algún tipo concreto de explorador para señalar datos históricos. Puesto que currentData hereda de Scanner, el modelo también necesita Scanner.

Sin embargo, sobre la base de las directrices de modelado y de las necesidades de las aplicaciones PM existentes y actuales, en este momento no se necesitan todas las capacidades definidas para esos objetos gestionados en las Recomendaciones UIT-T X.739 y Q.822. Por consiguiente, algunos de los lotes facultativos no se han incluido en este modelo. En el futuro, si surge la necesidad de alguna capacidad, se volverá a pensar en la posibilidad de incluir uno o más lotes pertinentes.

Para la mayoría de los objetos gestionados señalados anteriormente, su modelo IDL se puede derivar, mediante una traducción, de las directrices GDMO X.739 y Q.822. No obstante, para historyData, la traducción directa puede dar lugar a una gran cantidad de objetos gestionados. Estas entidades son almacenes de datos históricos de sólo lectura y no es necesario que sean interfaces CORBA. En lugar de ello se propone que se definan como valuetypes CORBA. La traducción de Q.822 definirá un valuetype de base para historyData, denominado HistoryDataValueType. Este valuetype HistoryDataValueType será accesible mediante métodos en la interfaz CurrentData. Cuando se cree una subclase de datos actuales se pretende que a la vez se cree una subclase del correspondiente HistoryDataValueType.

Puesto que los datos que se han de almacenar no están en interfaces CORBA sino en HistoryDataValueTypes, el informe de datos históricos mediante una simple traducción de exploradores homogéneos y sencillos GDMO no funciona. En lugar de ello, se necesita definir un nuevo tipo de explorador – uno que recopile datos de los HistoryDataValueTypes. En este modelo se trata de mantener las interfaces fuertemente tipificadas, siempre que sea posible. Esto se debe a que la recopilación de datos de tráfico es computacionalmente intensa, ya que el gestor de red debe recopilar y procesar muchos datos en un breve periodo de tiempo.

A este fin, HistoryDataScanner está concebido para recuperar datos históricos salvaguardados en objetos CurrentData pertinentes. En HistoryDataScanner se utiliza un mecanismo de transferencia de ficheros para recuperar grandes cantidades de datos puesto que cualquier otra técnica es probablemente más lenta y consume más recursos de computación y comunicación. El mecanismo recomendado soporta la capacidad de controlar la generación del fichero, la selección del formato del fichero y la notificación al usuario una vez producido el fichero, junto con la información necesaria para obtener el fichero. Sin embargo, el mecanismo concreto de transferencia de ficheros y la administración concreta de los ficheros están fuera del ámbito de esta Recomendación.

4.2.2 Modelo UML

En esta cláusula se presenta el modelo UML para la gestión de la calidad de funcionamiento. El modelo abarca Scanner, CurrentData, HistoryData (representados como el tipo HistoryDataValueType), ThresholdData e HistoryDataScanner. Como el modelo completo no cabe en una sola página, se presenta en las figuras 2, 3 y 4.

El modelo resultante descrito en UML se muestra en la siguiente cláusula. Los diagramas UML sólo muestran el modelo en formato de granularidad fina, pero estos diagramas se aplican a los formatos de granularidad fina y también a los de fachada. La interfaz IDL de fachada se traduce sistemáticamente de la interfaz IDL de granularidad fina como se define en la Rec. UIT-T X.780.1.

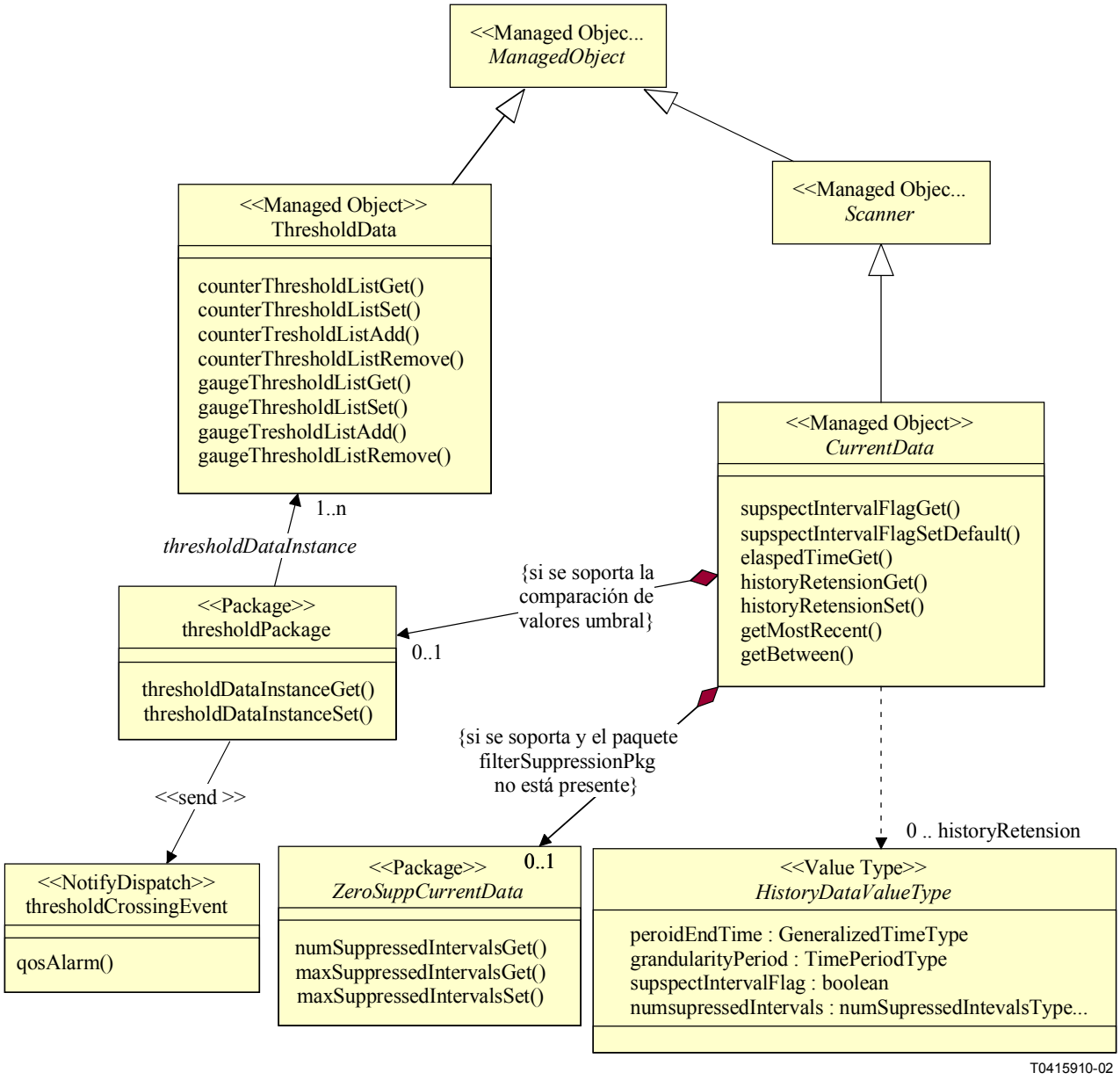

Figura 2/Q.822.1 – Diagrama de clases para CurrentData, HistoryData y ThresholdData

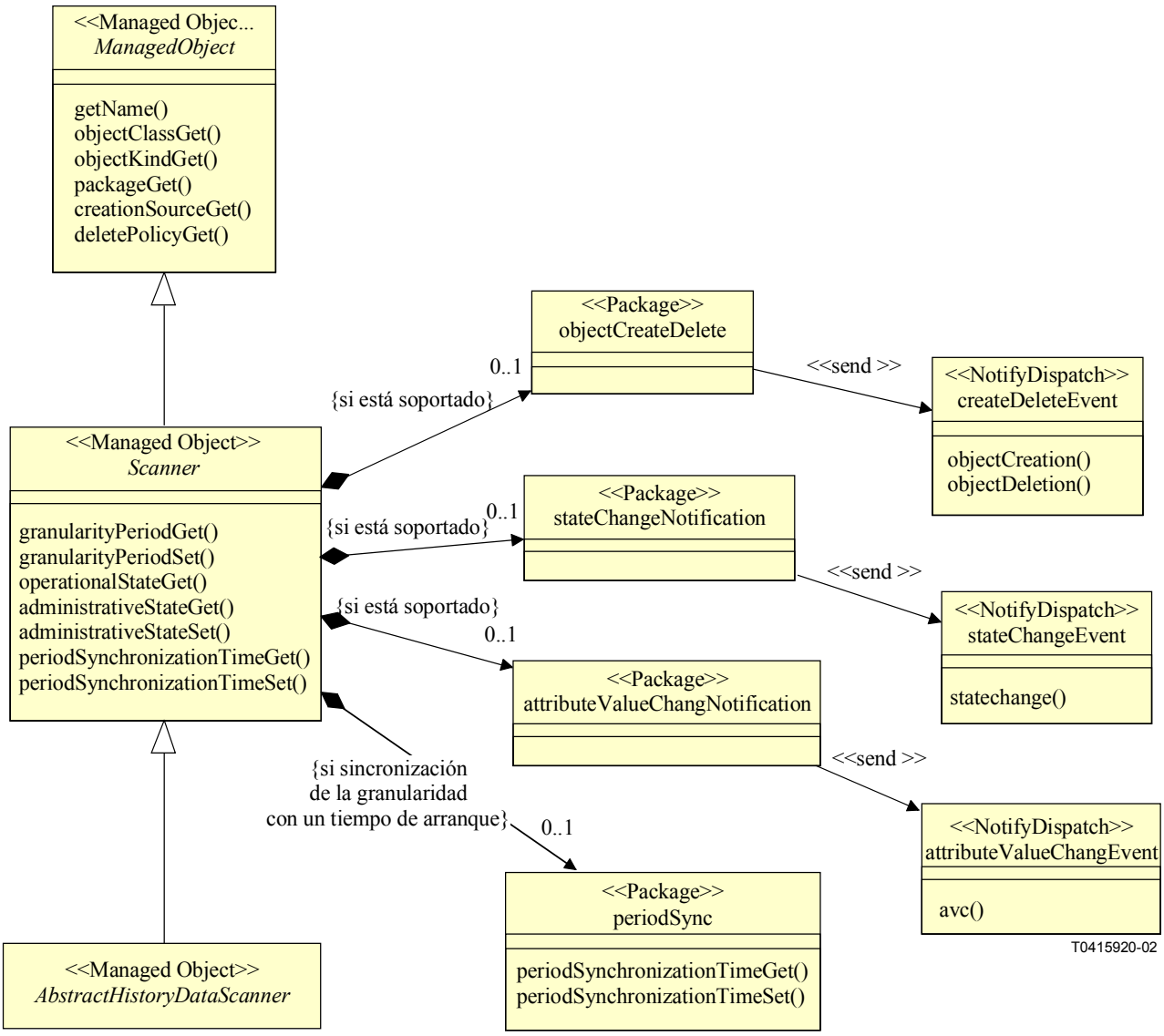

Figura 3/Q.822.1 – Diagrama de clases para Scanner

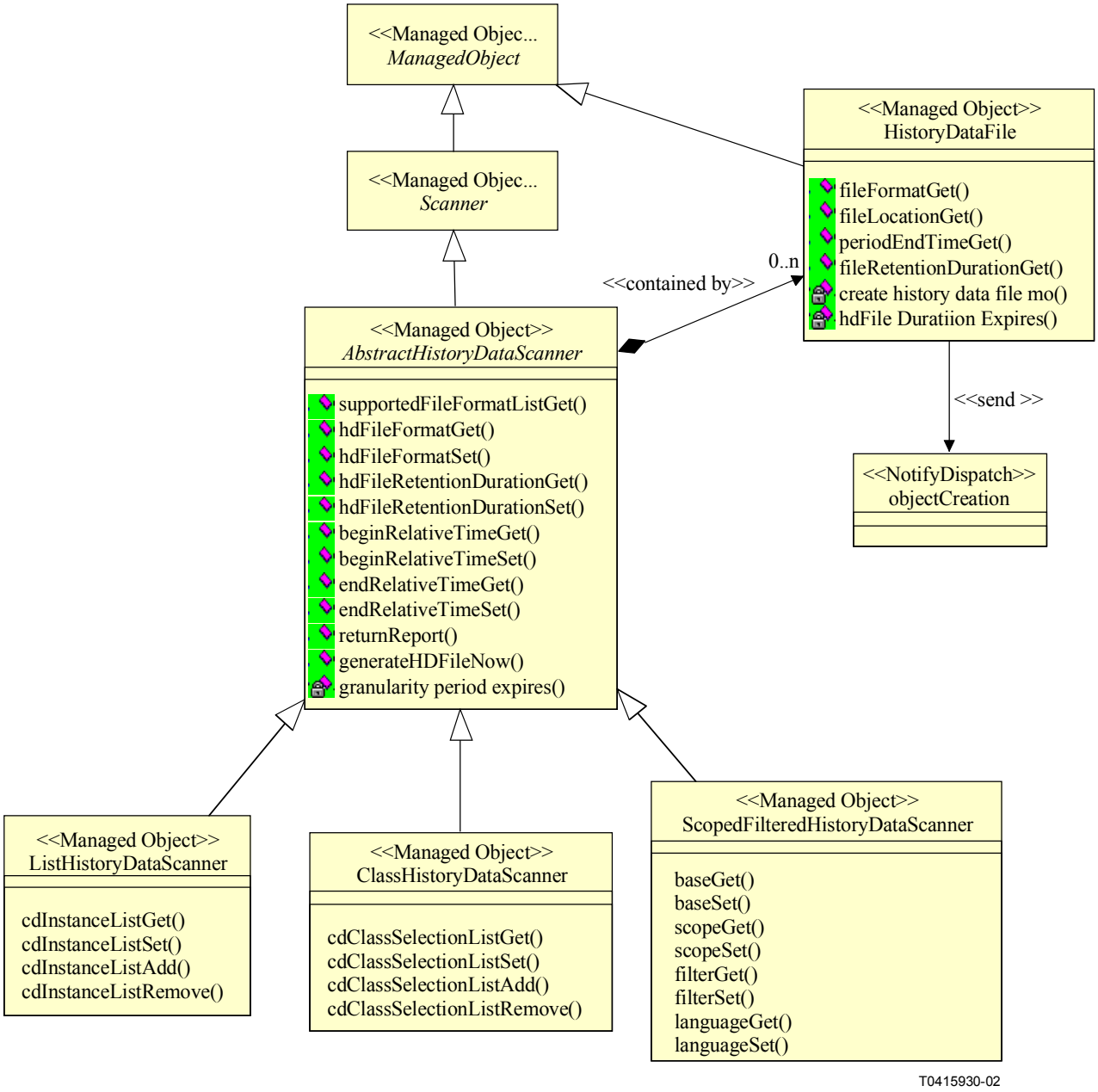

Figura 4/Q.822.1 – Diagrama de clases para HistoryDataScanner e HistoryDataFile

4.2.3 Relación de contenencia

En esta cláusula se muestra la relación de contenencia de las interfaces CORBA definidas en la presente Recomendación.

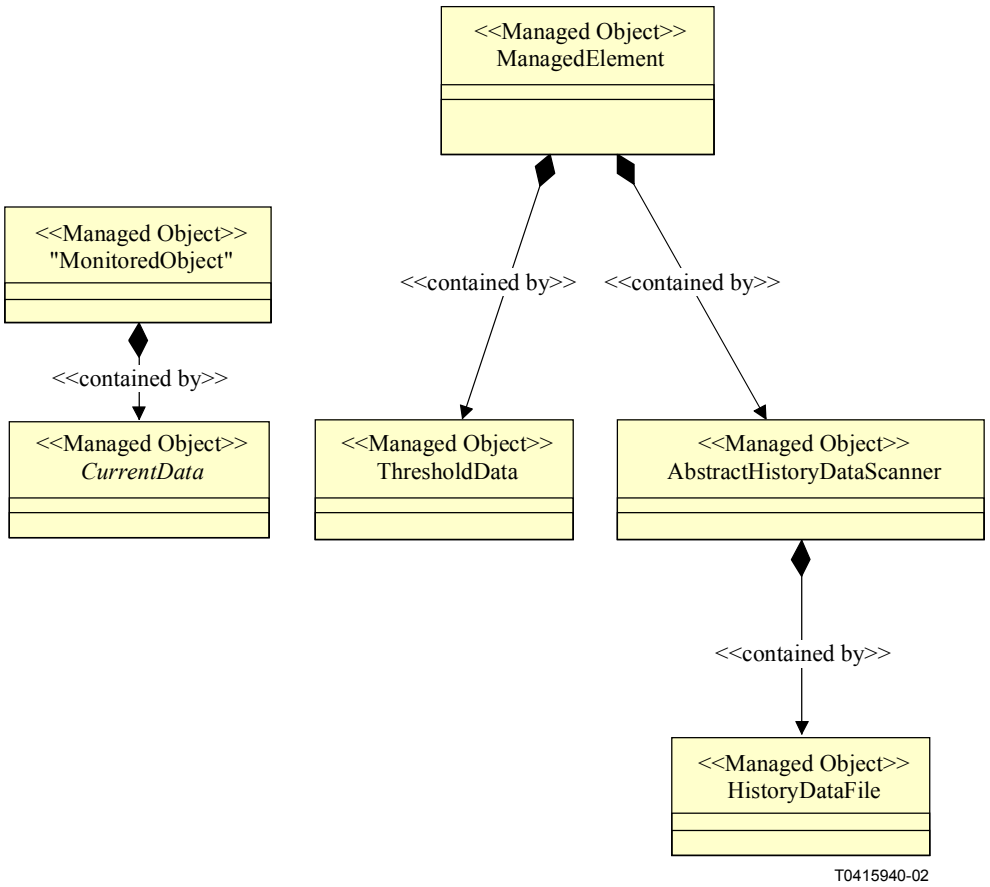

Figura 5/Q.822.1 – Relación de contenencia

4.3 Descripción de la interfaz de servicio

4.3.1 Scanner (explorador)

Un objeto gestionado de esta interfaz representa la aptitud para recuperar valores de atributos de objetos gestionados y producir información sumaria a partir de esos valores. Esta información sumaria puede hacerse disponible en atributos, notificaciones, respuestas a acciones o una combinación de éstos. La información sumaria puede consistir en valores de atributos observados o estadísticas calculadas a partir de estos valores (con respecto al tiempo o con respecto a objetos gestionados).

Los valores de atributos observados se recuperan durante una "exploración" ("scan"), que se inicia periódicamente al final de cada periodo de granularidad, siempre y cuando el periodo de granularidad no sea cero.

El atributo periodo de granularidad indica la longitud del periodo de granularidad. El periodo de granularidad en el objeto explorador no deberá modificarse a menos que el valor del estado administrativo sea 'bloqueado' ('locked'). Si no está presente el lote sincronización de periodos, el momento en el cual se inicia el periodo de granularidad después de que se haya desbloqueado el explorador es un asunto local.

El atributo estado administrativo se utiliza para suspender o reanudar la función de exploración. Si el estado administrativo tiene el valor 'desbloqueado' ('unlocked'), el explorador tiene el permiso administrativo para realizar exploraciones. Si el estado administrativo tiene el valor 'locked', el explorador tiene la prohibición administrativa de realizar exploraciones.

El atributo estado operacional representa la aptitud operacional del explorador para realizar sus funciones.

Si el atributo lote notificación de cambio de valor está presente, los cambios en el periodo de granularidad o en el tiempo de sincronización de periodos provocarán la emisión de notificaciones de cambio de valor atributo. Si el lote notificación de cambio de estado está presente, los cambios en el estado operacional o en el estado administrativo provocarán la emisión de notificaciones de cambio de estado. Si el lote notificación de creación/supresión de objeto está presente, la creación o supresión de un objeto de la subclase del explorador provocará la emisión de la correspondiente notificación de creación o supresión.

En la entidad están definidos los siguientes atributos:

- *administrativeState* – se utiliza para activar y desactivar (suspender o reanudar administrativamente) la función que realiza el explorador.
- *granularityPeriod* – especifica el tiempo durante el cual se realiza el "scan" (la exploración).
- *operationalState* – determina si la función de exploración representada por este objeto puede operar normalmente.

El lote periodSynchronizationPackage está presente si se requiere sincronización interna de agente configurable, de periodos de tiempo repetitivos.

El siguiente atributo se define en el periodSynchronizationPackage:

- *periodSynchronizationTime* – contiene el tiempo con el cual se sincroniza el periodo de tiempo repetitivo. El comienzo de cada periodo es un número entero de periodos antes o después del tiempo especificado por este atributo.

La interfaz Scanner no es ejemplificable.

4.3.2 CurrentData

4.3.2.1 CurrentData

La clase de objeto CurrentData, heredada de scanner, es una clase de objetos de soporte gestionados que registran los datos de calidad de funcionamiento actuales para propósitos de supervisión. Los datos de calidad de funcionamiento (resultados de mediciones) de recursos supervisados se modelan como atributos en la definición de las subclases de CurrentData, puesto que CurrentData no puede ejemplificarse. Los objetos que representan recursos supervisados contienen los objetos CurrentData correspondientes en las relaciones de vinculación de nombre.

Los resultados de las mediciones se recopilan durante un intervalo de tiempo (por ejemplo, de 5 minutos) especificado por el atributo granularityTime. Al final de cada intervalo, el atributo elapsedTime se actualizará a la diferencia entre el tiempo actual y el inicio del intervalo de supervisión presente. Los resultados de las mediciones, granularityTime y suspectIntervalFlag, se copian en una estructura de datos históricos correspondiente (representada como HistoryDataValueType). Además, el periodEndTime de la estructura de datos históricos se asigna al tiempo actual.

Si historyRetention es mayor que cero, las estructuras de datos históricos se retendrán en el sistema gestionado durante un lapso al menos equivalente al número de intervalos especificado en el atributo historyRetention. Durante ese tiempo, los datos históricos son accesibles a través de un conjunto de operaciones:

- *getMostRecent* – recupera los datos históricos más recientes. Por "más reciente" ha de entenderse el último registro de datos históricos salvaguardado, que puede estar varios intervalos antes del tiempo actual porque el historyRetention puede estar puesto a cero o todos los resultados de mediciones recopilados pueden ser cero. Es posible que getMostRecent no pueda recuperar ningún dato. Por ejemplo, si todos los resultados de

mediciones son cero (no hay datos históricos salvaguardados), o historyRetention ha sido puesta a cero. El valor booleano devuelto por el método getMostRecent indica la disponibilidad de los datos. Si este valor es verdadero significa que el parámetro out contiene los datos, y si es falso el parámetro está indefinido.

- *getBetween* – recupera un conjunto de datos históricos dentro de una ventana de tiempo especificada por los parámetros de comienzo y fin de tiempo. El resultIterator se utiliza para transferir grandes bloques de datos. Si howMany no se proporciona, la cantidad inicial de datos devueltos es un asunto local.

En esta entidad se definen los siguientes atributos:

- *suspectIntervalFlag* – se utiliza para indicar que los datos de calidad de funcionamiento para el periodo actual pueden no ser fiables. Son razones para que esto suceda:
  - El recurso que efectivamente está realizando la recopilación de datos detectó datos sospechosos.
  - Se ha producido una transición del atributo administrativeState hacia o desde el estado 'lock'.
  - Se ha producido una transición del operationalState hacia o desde el estado 'disabled'.
  - Los contadores de la calidad de funcionamiento se pusieron a cero en el curso del intervalo.
  - El ejemplar de objeto CurrentData (o una subclase del mismo) se creó durante el periodo de supervisión.
- *elapsedTime* – representa la diferencia entre el tiempo actual y el comienzo del intervalo de supervisión presente.
- *historyRetention* – especifica el número mínimo de intervalos que la estructura de datos históricos (recién creada) debe preservar.

Si thresholdPackage está presente, el objeto CurrentData contiene como mínimo un puntero a un objeto ThresholdData. Si se sobrepasa cualquiera de los umbrales (definidos en el objeto ThresholdData a que se hace referencia), el objeto CurrentData emitirá una notificación de alarma de calidad del servicio (QoS, *quality of service*). El campo thresholdInfo de la alarma contendrá el atributo de medición que sobrepasa el umbral. Si se producen uno o más rebasamientos adicionales de umbral durante el intervalo de tiempo actual, se emitirán alarmas adicionales.

El atributo thresholdDataInstanceList contiene un conjunto de punteros a objetos ThresholdData, especificados por el sistema o los sistemas de gestión. Por lo tanto, para un mismo atributo supervisado se pueden emitir más de una notificación de alarma QoS. Por otro lado, es posible que las notificaciones no concuerden: por ejemplo, una notificación indica que la alarma está activada, y la otra que está desactivada. Es responsabilidad de los sistemas de gestión mantener la coherencia de las notificaciones.

Deberán tener efecto inmediato los nuevos umbrales que resulten de la modificación del atributo thresholdDataInstanceList o del cambio de un valor de umbral en el objeto ThresholdData a que se hace referencia. Si la condición de alarma existe antes de que se produzca un cambio en el valor umbral (es decir, se sobrepasó un valor umbral anterior) y el nuevo valor umbral está fuera de la gama del antiguo valor umbral (por ejemplo, en el caso de un contador ascendente, el nuevo valor umbral es mayor que el antiguo valor de umbral), y el valor actual de la medición está dentro de la gama admisible del nuevo valor umbral, entonces se emite una notificación de alarma QoS con una severidad de 'clear'. Si el nuevo valor de umbral se establece dentro de la gama del antiguo valor umbral, de manera que el nuevo valor umbral también es sobrepasado, se emite una notificación de alarma QoS si una condición de alarma no está ya pendiente.

En el thresholdPackage se define el siguiente atributo:

- *thresholdDataInstanceList* – es un atributo 'list' en el cual cada ítem es un puntero a un objeto ThresholdData que contiene límites de umbral para parámetros de calidad de funcionamiento.

El zeroSuppressionPackage es un lote condicional de CurrentData. Cuando este lote está presente y el intervalo termina con mediciones de calidad de funcionamiento 'todos a ceros', no se crea una estructura de datos históricos.

En el zeroSuppressionPackage se definen los atributos siguientes:

- *numSuppressedIntervals* – se utiliza para contar el número de intervalos consecutivos para los que se ha producido una supresión (es decir, no se ha creado una estructura de datos históricos). Este atributo refleja mediciones de calidad de funcionamiento hasta, pero sin incluir, el intervalo actual. Este atributo se incrementa al final de un intervalo si se ha producido una supresión. En caso contrario, el atributo se reinicializa. Si su valor llega a alcanzar maxSuppressedIntervals, se creará un registro de datos históricos, que será reinicializado.
- *maxSuppressedIntervals* – limita el número máximo de intervalos suprimidos que pueden recopilarse sin crear una estructura de datos históricos. El valor por defecto −1 significa que no hay límite al número de intervalos suprimidos consecutivos. Por otro lado, el maxSuppressedIntervals es efectivamente igual a infinito.

Por ejemplo, supóngase un ejemplar de (una subclase de) CurrentData de supresión de ceros con maxSuppressedIntervals fijado a 32, y el intervalo fijado a 15 minutos. Para la compresión de registros, esto significa que después de 32 intervalos (8 horas) consecutivos, suprimidos (por ejemplo, todos ceros) al menos un registro de datos históricos (con parámetros PM todos ceros) se generará con una cuenta de 32. Esto garantiza que se creará al menos un registro de datos históricos por maxSuppressedIntervals.

La interfaz CurrentData no es ejemplificable.

4.3.2.2 HistoryData

HistoryData, representado como HistoryDataValueType, contiene una copia de las mediciones de calidad de funcionamiento y otros atributos seleccionados que están presentes en un objeto de datos actual al final del intervalo actual (por ejemplo, 5 minutos). Al final de cada intervalo se crea un nuevo registro de esta estructura valuetype si el atributo historyRetention en el objeto de datos actual es mayor que cero. Este registro se retendrá entonces en el elemento de red durante al menos un periodo de tiempo equivalente al número de intervalos especificados en el atributo historyRetention.

Este valuetype genérico modela los datos históricos. Los objetos de datos actuales específicos (por ejemplo, los objetos de datos actuales SDH/SONET) se definen con valuetypes históricos específicos que se heredan de este valuetype HistoryDataValuetype genérico.

En la entidad se definen los atributos siguientes:

- *periodEndTime* – indica el tiempo al final del intervalo.
- *granularityPeriod* – se utiliza para copiar el mismo atributo del objeto CurrentData.
- *suspectIntervalFlag* – se utiliza para copiar el mismo atributo del objeto CurrentData.
- *numSupressedIntervals* – indica el número de intervalos suprimidos que se han producido antes de la creación del registro valuetype histórico. Tiene un valor por defecto de 0. Cuando se copia del objeto CurrentData se utiliza el valor por defecto. Cuando se copia de CurrentData con ZeroSuppressionPackage se utiliza el valor de atributo correspondiente.

4.3.3 ThresholdData

La clase de objeto ThresholdData es una clase de objetos de soporte gestionados que contienen los valores de umbral establecidos para los parámetros PM. Debe haberse creado un ejemplar de al menos uno de los lotes counterThresholdListPackage o gaugeThresholdListPackage.

Se establecen umbrales mediante la utilización de la clase de objeto gestionado ThresholdData. Los objetos ThresholdData especifican los umbrales que se aplican. Los valores umbral en un objeto ThresholdData pueden aplicarse a múltiples CurrentData y a los objetos ThresholdData apuntan objetos CurrentData. El objeto CurrentData (o sus subclases) indican el intervalo de recopilación que se utiliza con el umbral. Siempre que se rebasa un umbral se genera una notificación qualityOfServiceAlarm.

Puede especificarse un umbral para un solo objeto gestionado. En este caso, el objeto ThresholdData está contenido en el objeto gestionado que se está observando. Un objeto CurrentData dentro del objeto gestionado apunta al objeto ThresholdData. Esta vinculación es necesaria para asegurar la coherencia en caso de que hayan múltiples objetos gestionados.

A veces conviene que un umbral se aplique a un grupo de objetos gestionados. En este caso, el objeto ThresholdData es externo al objeto gestionado que se está observando. Puede estar contenido dentro del objeto ManagedElement. Al objeto ThresholdData apuntan todos los objetos CurrentData a los cuales se aplica.

Si el lote createDeleteNotificationsPackage está presente, la creación o supresión de un objeto de umbral provocará la emisión de una notificación objectCreation u objectDeletion.

Si el atributo ValueChangeNotificationPackage está presente, cualquier cambio de umbral provocará la emisión de una notificación de attributeValueChange.

El counterThresholdListPackage está presente si un ejemplar lo soporta y el gaugeThresholdListPackage no está presente.

En el counterThresholdListPackage se define el atributo siguiente:

- *counterThresholdList* – contiene un conjunto de configuraciones de umbrales para atributos de calidad de funcionamiento del tipo contador (por ejemplo, segundos con error). Cada configuración de umbrales consiste en el identificador de atributo, el valor de umbral y (facultativamente) la severidad del suceso de rebasamiento de umbral.

El gaugeThresholdListPackage está presente si un ejemplar lo soporta y el counterThresholdListPackage no está presente.

En el gauge ThresholdPackage se define el atributo siguiente:

- *gaugeThresholdList* – contiene un conjunto de configuraciones de umbrales para atributos de calidad de funcionamiento del tipo calibre (*gauge*). Cada configuración de umbrales consiste en el identificador de atributo, el valor de umbral y (facultativamente) la severidad del suceso de rebasamiento de umbral.

4.3.4 HistoryDataFile

HistoryDataFile es una subclase de ManagedObject. Establece una correspondencia biunívoca con un fichero de datos históricos. Cada fichero de datos históricos está asociado a un objeto HistoryDataFile. Cada objeto HistoryDataFile tiene un fichero de datos históricos subyacente. El HistoryDataFile sólo se crea después de que se haya producido el archivo de datos históricos. El objeto HistoryDataFile se suprime inmediatamente después de haberse purgado el fichero de datos históricos.

El sistema gestionado controlará el ciclo de vida del HistoryDataFile. Sin embargo, se proporciona un comportamiento de destruir, que consiste en que el fichero de datos históricos es purgado si se suprime el objeto HistoryDataFile, cuando el sistema de gestión quiere limpiar el fichero de datos históricos antes de la expiración del periodo de retención del fichero.

El objeto HistoryDataFile tiene los atributos siguientes:

- *fileLocation* – identifica la ubicación del fichero en formato URL. El tiempo que ha de transcurrir antes de reutilizar la misma fileLocation para dos ficheros diferentes debe ser ampliamente suficiente para evitar cualquier confusión.
- *fileFormat* – indica el formato de fichero utilizado.
- *periodEndTime* – indica el tiempo (la hora) al final de la generación del fichero.
- *fileRetentionDuration* – indica el periodo de retención del fichero antes de que el fichero sea purgado debido a la recopilación de datos corrompidos del sistema gestionado.

Cuando se genera un fichero de datos históricos y se crea un objeto HistoryDataFile, el sistema gestionado enviará una notificación objectCreation para informar al sistema de gestión la disponibilidad del archivo de datos históricos. Cuando se suprime un fichero de datos históricos y se borra el objeto HistoryDataFile asociado, el sistema gestionado podría enviar una notificación objectDeletion al sistema de gestión si el paquete objectDeleteNotificationPackage está soportado.

Aunque exista un HistoryDataFileFactory, el sistema de gestión nunca creará un objeto HistoryDataFile. La HistoryDataFileFactory está destinada a ser utilizada por un sistema gestionado.

4.3.5 AbstractHistoryDataScanner

AbstractHistoryDataScanner es una subclase de Scanner y una superclase de todos los demás exploradores de datos históricos. Es la colección de atributos y operaciones comunes de los exploradores de datos históricos. El AbstractHistoryDataScanner no es ejemplificable.

Los exploradores de datos históricos soportan la exploración y señalación de datos históricos proporcionando la aptitud para controlar la generación del fichero de datos históricos, la selección del formato del fichero y la notificación al usuario cuando se produce el fichero.

El objeto AbstractHistoryDataScanner tiene los atributos siguientes:

- *supportedFileFormatList* – indica los formatos de fichero soportados en el sistema gestionado, puesto que puede soportar más de un formato de fichero.
- *hdFileFormat* – indica el tipo de formato que habrá de utilizarse.
- *fileRetentionDuration* – indica el periodo de retención del fichero para todos los ficheros de datos históricos generados por este explorador.
- *beginRelativeTime* – indica el tiempo (hora) de comienzo relativo al tiempo actual junto con el tiempo relativo de terminación definiendo una ventana para seleccionar datos históricos. Un valor negativo del tiempo relativo de comienzo significa tiempo pasado.
- *endRelativeTime* – indica el tiempo (hora) de terminación relativo al tiempo actual. Un valor negativo del tiempo relativo de terminación significa tiempo pasado. Un valor cero significa que no hay tiempo de terminación lo cual entrañaría la generación periódica de fichero hasta que el sistema de gestión detuviera el proceso.
- *granularityPeriod (heredado de Scanner)* – establece una ventana de tiempo de manera que los datos históricos cuyas indicaciones de tiempo estén dentro de esta ventana se informarán en un fichero.

Antes de dar cualquier valor a estos atributos, el administrativeState se ha de poner a 'lock', y cuando el administrativeState está puesto a 'unlock' comienza la generación de archivo periódica. Si el periodo de granularidad es cero, no hay generación de archivo periódica.

Al final de cada periodo de granularidad del explorador, el explorador de datos históricos recorre los datos históricos desde un conjunto de objetos de datos actuales especificados en el explorador detallado, reúne todos los datos cuyo tiempo de terminación de periodo está dentro del actual intervalo de tiempo de granularidad, y genera un fichero de datos en el formato indicado en hdFileFormat. Cuando se produce el fichero, el sistema gestionado crea un objeto HistoryDataFile y se envía una notificación objectCreation al sistema de gestión. La fuente de la notificación objectCreation es el objeto HistoryDataFile recién creado.

El explorador de datos históricos dispone de dos operaciones para informar datos históricos no periódicos:

- *returnHDReport* – retorna una secuencia de tipos de valor de datos históricos cuyo tiempo de terminación de periodo está entre el momento actual y el final del último periodo de granularidad del explorador.
- *generateHDFileNow* – genera un fichero de datos históricos inmediatamente en lugar de esperar hasta el siguiente periodo de granularidad del explorador. El tiempo de terminación de periodo del tipo de valor de datos históricos en este fichero estará entre el momento actual y el final del último periodo de granularidad.

4.3.6 ListHistoryDataScanner

El ListHistoryDataScanner es un explorador de datos históricos especializado y es una subclase de AbstractHistoryDataScanner. Retorna informes y ficheros de datos históricos sobre la base de una lista de objetos simples.

El objeto ListHistoryDataScanner tiene el siguiente atributo:

- *cdInstancesList* – contiene un conjunto de objetos de datos actuales cuyos datos históricos deben ser explorados/informados a un fichero.

La figura 6 describe un caso de utilización sencillo de ListHistoryDataScanner:

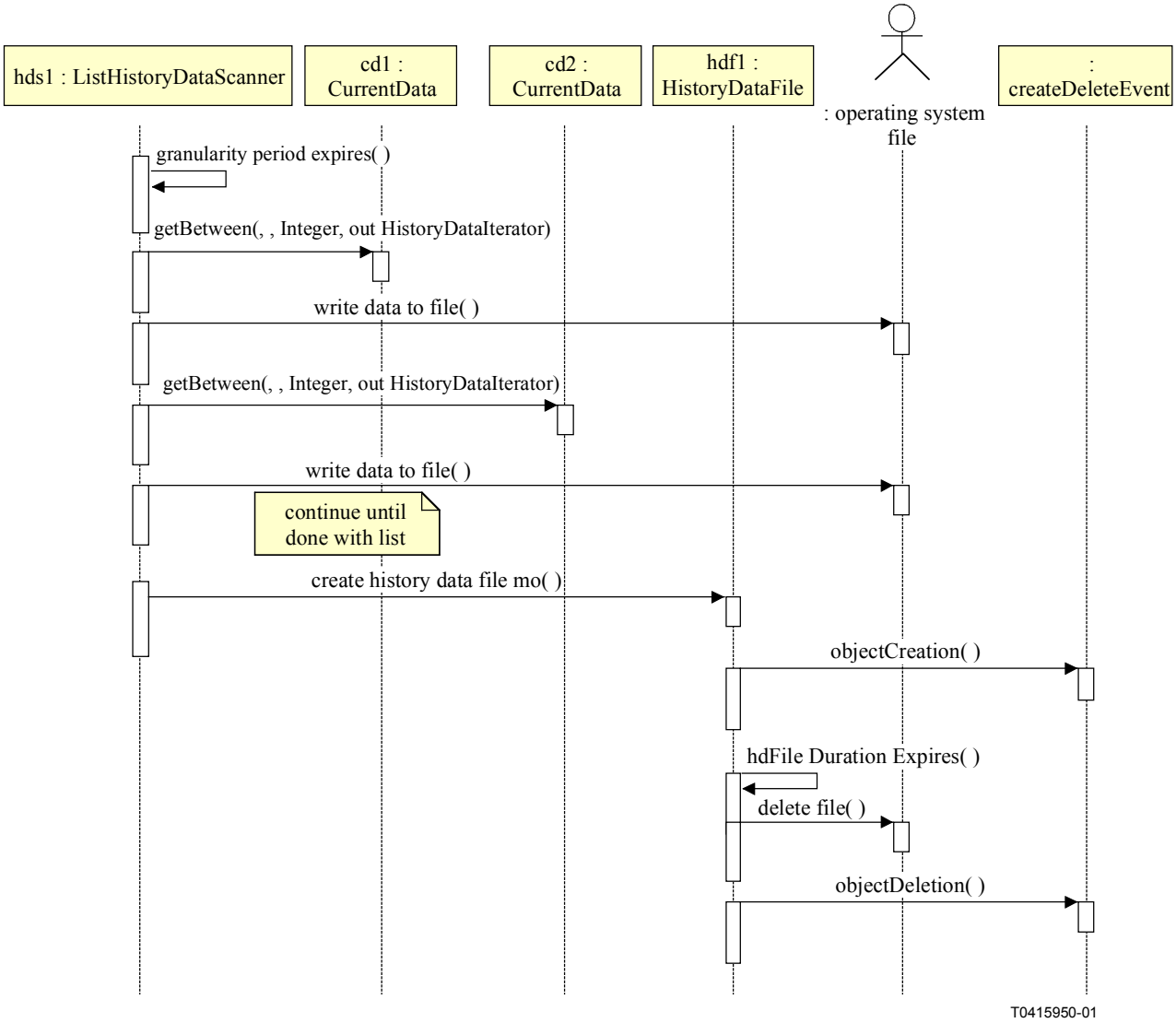

Figura 6/Q.822.1 – Un caso de utilización de ListHistoryDataScanner

4.3.7 ClassHistoryDataScanner

El ClassHistoryDataScanner es un explorador de datos históricos especializado y es una subclase de AbstractHistoryDataScanner. Retorna informes y ficheros de datos históricos sobre la base de una lista de clases de objetos.

El objeto ClassHistoryDataScanner tiene el atributo siguiente:

- *cdClassSelectionList* – contiene un conjunto de clases de datos actuales agrupadas por el ManagedObject contenedor (por ejemplo, equipo de red). Los objetos de datos actuales cuyos datos históricos deben ser explorados/informados a un fichero son aquellos ejemplares de las clases de datos actuales especificadas contenidos por el respectivo ManagedObject.

Antes de generar un nuevo informe o fichero, ClassHistoryDataScanner reevalúa los ejemplares de datos actuales concordantes para cualquier ejemplar de datos actuales añadido o suprimido desde la última exploración.

4.3.8 ScopedFilteredHistoryDataScanner

El ScopedFilteredHistoryDataScanner es un explorador de datos históricos especializado y es una subclase de AbstractHistoryDataScanner. Retorna informes y ficheros de datos históricos sobre la base de una determinación del alcance y un filtrado.

El objeto ScopedFilteredHistoryDataScanner tiene los atributos siguientes:

- *base* – objeto gestionado de base utilizado para la determinación del alcance.
- *scope* – valor del alcance según está definido en la Rec. UIT-T Q.816.
- *filter* – expresión de filtro de constricción que se utiliza para evaluar los objetos concordantes.
- *language* – una cadena que indica el lenguaje en que está escrita la expresión de filtro.

Los objetos de datos actuales cuyos datos históricos deben ser explorados/informados a un fichero son aquellos objetos de datos actuales seleccionados mediante la operación de determinación de alcance y filtrado. Antes de generar un nuevo informe o fichero, el ScopedFilteredHistoryDataScanner vuelve a aplicar la operación de determinación de alcance y filtrado para seleccionar el último conjunto de objetos de datos actuales.

5 IDL del servicio de gestión de la calidad de funcionamiento

```
#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"


/**
El presente código IDL deberá almacenarse en un archivo denominado
"itut_q822_1.idl" situado en el directorio de búsqueda que utilizan los
compiladores IDL en su sistema.
*/


/**
El presente módulo, itut_q822d1, contiene la definición IDL sobre la base de
objetos definidos en la Rec. UIT-T Q.822. Las definiciones de IDL en este archivo
son las interfaces objeto.
*/
module itut_q822d1
{


/**
```

5.1 Imports (importaciones)

```
*/

/**
```

```
Tipos importados de 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;


/**
Tipos importados de itut_x780ct
*/
     typedef itut_x780ct::SeverityIndicatingThresholdType
          SeverityIndicatingThresholdType;
     typedef itut_x780ct::SeverityIndicatingGaugeThresholdSetType
          SeverityIndicatingGaugeThresholdSetType;
     typedef itut_x780ct::TimeIntervalType TimeIntervalType;
     typedef itut_x780ct::TimePeriodType TimePeriodType;


/**
Tipos importados de itut_q816
*/
     typedef itut_q816::ScopeType ScopeType;
     typedef itut_q816::FilterType FilterType;
     typedef itut_q816::LanguageType LanguageType;


/**
Tipos importados de itut_m3120
*/
     typedef itut_m3120::ManagedElementNameType ManagedElementNameType;
/**
Interfaces importadas from itut_x780

itut_x780::ManagedObject
itut_x780::ManagedObjectFactory
*/


/**
```

5.2 Forward Declarations (declaraciones hacia adelante)

```
*/

/**
Declaraciones directas de interfaz
*/
     interface Scanner;
     interface HistoryDataIterator;
     interface CurrentData;
     interface ThresholdData;
     interface HDFile;
     interface AbstractHDScanner;
```

```
     interface ListHDScanner;
     interface ClassHDScanner;
     interface ScopedFiltedHDScanner;


/**
Declaraciones directas de valor de tipo
*/
     valuetype ScannerValueType;
     valuetype CurrentDataValueType;
     valuetype HistoryDataValueType;
     valuetype ThresholdDataValueType;
     valuetype HDFileValueType;
     valuetype AbstractHDScannerValueType;
     valuetype ListHDScannerValueType;
     valuetype ClassHDScannerValueType;
     valuetype ScopedFiltedHDScannerValueType;


/**
Declaraciones directas de interfaz
*/
     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 (estructuras y definiciones de tipo)

```
*/

/**
Este tipo de datos define una secuencia de HistoryDataValueType. El orden es
importante.
*/
     typedef sequence <HistoryDataValueType> HistoryDataSeqValueType;

     typedef sequence <CurrentDataNameType> CurrentDataNameSetType;

     typedef UIDType HDFileFormatType;

     typedef sequence <HDFileFormatType> HDFileFormatSetType;

     typedef Istring URLType;

     struct CounterThresholdSettingType
     {
          string                          attributeName;
          SeverityIndicatingThresholdType  threshold;
     };

/**
El orden no es importante.
*/
     typedef sequence <CounterThresholdSettingType> CounterThresholdSetType;

     struct GaugeThresholdSettingType
```

```
     {
          string                        attributeName;
          SeverityIndicatingGaugeThresholdSetType threshold;
     };

/**
El orden no es importante.
*/
     typedef sequence<GaugeThresholdSettingType>
          GaugeThresholdSetType;

/**
Esta estructura de datos representa todas las clases CurrentData contenidas en el
objeto de continencia a través de una relación de continencia directa o
indirecta.
*/
     struct CDClassSelectionType
     {
          MONameType          containingObject;
          ScopedNameSetType        cdClassList;
     };

     typedef sequence<CDClassSelectionType> CDClassSelectionSetType;


/**
```

5.4 Exceptions (excepciones)

```
*/

/**
Excepciones y constantes para el paquete condicional
*/
     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";


/**
Excepciones para la gestión de la calidad
*/


/**
```

5.5 Interfaces – Fine-grained (interfaces – granularidad fina)

```
*/

/**
```

5.5.1 Scanner

```
Para obtener una explicación del funcionamiento véase 4.3.1.

Esta interfaz contiene los siguientes PAQUETES CONDICIONALES.

- periodSynchronizationPackage: está presente si se necesita una sincronización
interna de agente configurable de periodos de tiempo repetitivos.

La interfaz Scanner no es ejemplificable.
*/
     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

```
Para obtener una explicación del funcionamiento véase 4.3.2.1.

Esta interfaz contiene los siguientes PAQUETES CONDICIONALES.

- thresholdPackage: presente si se necesita emitir una notificación de alarma de
calidad del servicio cuando se sobrepasa un umbral.

- zeroSupressionPackage: presente si un ejemplar lo soporta.

La interfaz CurrentData no es ejemplificable.
*/

/**
```

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

```
Para obtener una explicación del funcionamiento véase 4.3.2.2.
*/
     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 se utiliza para la transferencia iterativa de grandes bloques
de datos anteriores.
*/
     interface HistoryDataIterator
     {
/**
Este método se utiliza para devolver el número siguiente de datos históricos. howMany es
el número máximo de datos anteriores para los cuales se deberán devolver los resultados en
el primer bloque, y no deben ser cero. historyDataList es una secuencia de registros de
datos anteriores.
El método devuelve el valor verdadero si hay datos en el parámetro de salida y
falso en caso contrario.
*/
          boolean getNext
               (in unsigned short howMany,
               out HistoryDataSeqValueType historyDataList)
               raises (itut_x780::ApplicationError);

/**
Este método se utiliza para destruir el iterador y liberar sus recursos. Esto lo
debe realizar la aplicación.
*/
          void destroy ();

     };  // interface HistoryDataIterator


/**
```

5.5.2.4 CurrentData

```
*/
     interface CurrentData: Scanner
     {
/**
Define el valor por omisión de suspectIntervalFlag
*/
          const boolean suspectIntervalFlagDefault = FALSE;

/**
```

```
Define el valor por omisión de maxSuppressedIntervals. Si este valor es -1
significa que no hay límite en el número de intervalos consecutivos suprimidos.
Es decir, el maxSuppressedIntervals es igual a infinito.
*/
          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

```
Para una explicación de su funcionamiento, véase 4.3.3.

Esta interfaz contiene los siguientes PAQUETES CONDICIONALES.

- counterThresholdListPackage: está presente si un ejemplar lo soporta y el
paquete gaugeThresholdListPackage no está presente.

- gaugeThresholdListPackage: está presente si un ejemplar lo soporta y si el
paquete counterThresholdListPackage no está presente.
*/
     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)

```
Para obtener una explicación de su funcionamiento, véase 4.3.4.

Esta interfaz contiene los siguientes PAQUETES CONDICIONALES.

- objectDeleteNotificationPackage: está presente si un ejemplar lo soporta.
*/
    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)

```
Para tener una explicación de su funcionamiento, véase 4.3.5.
*/
    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);

/**
El hdFileFormat debe ser de uno de los formatos de archivo soportados.
*/
          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

```
Para obtener una explicación de su funcionamiento, véase 4.3.6.
*/
     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

```
Para obtener una explicación de su funcionamiento, véase 4.3.7.
*/
     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

```
Para obtener una explicación de su funcionamiento, véase 4.3.8.
*/
    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 (interfaces – fachada)

```
Los exploradores de datos anteriores de fachada se definen en la
Rec. UIT-T X.780.1. Sin embargo los exploradores de datos anteriores de
granularidad fina serían suficientes para la implementación de fachada.
*/

/**
```

5.6.1 Scanner_F

```
Para obtener una explicación de su funcionamiento, véase 4.3.1.

Esta interfaz contiene los siguientes PAQUETES CONDICIONALES.

- periodSynchronizationPackage: presente si se necesita una sincronización
interna de agente configurable de periodos de tiempo repetitivos.

La interfaz Scanner no es ejemplificable.
*/
     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

```
Para obtener una explicación sobre su funcionamiento, véase 4.3.2.1.

Esta interfaz contiene los siguientes PAQUETES CONDICIONALES.

- thresholdPackage: está presente si se necesita emitir una notificación de
alarma de calidad del servicio al sobrepasar umbrales.

- zeroSupressionPackage: está presente si un ejemplar lo soporta.

La interfaz CurrentData no es ejemplificable.
*/
     interface CurrentData_F: Scanner_F
     {
/**
Define el valor por omisión para suspectIntervalFlag
*/
          const boolean suspectIntervalFlagDefault = FALSE;

/**
Define el valor por omisión para maxSuppressedIntervals. Si este valor es igual
a -1 significa que no hay límite en el número de intervalos consecutivos
suprimidos. Es decir, maxSuppressedIntervals es igual a infinito.
*/
          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

```
Para obtener una explicación sobre su funcionamiento, véase 4.3.3.

Esta interfaz contiene los siguientes PAQUETES CONDICIONALES.

- counterThresholdListPackage: está presente si un ejemplar lo soporta y no está
presente gaugeThresholdListPackage.

- gaugeThresholdListPackage: está presente si un ejemplar lo soporta y no está
presente counterThresholdListPackage.
*/
    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)

Para obtener una explicación sobre su funcionamiento, véase 4.3.4.

Esta interfaz contiene los siguientes PAQUETES CONDICIONALES.

```
- objectDeleteNotificationPackage: está presente si un ejemplar lo soporta.
*/
     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)

```
Para obtener una explicación sobre su funcionamiento, véase 4.3.5.
*/
     interface AbstractHDScanner_F: Scanner_F
     {

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

/**
El hdFileFormat debe ser de uno de los formatos de archivos soportados.
*/
          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

```
Para obtener una explicación sobre su funcionamiento, véase 4.3.6.
*/
     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

```
Para obtener una explicación sobre su funcionamiento, véase 4.3.7.
*/
     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

```
Para obtener una explicación sobre su funcionamiento, véase 4.3.8.
*/
     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 (notificaciones)

```
En la presente Recomendación no se definen notificaciones específicas del modelo.
*/


/**
```

5.8 Name Binding (vinculación de nombre)

```
*/

/**
El siguiente módulo contiene la información de vinculación de nombres.
*/
     module NameBinding
     {


/**
```

5.8.1 ThresholdData

```
Esta vinculación de nombre se utiliza para denominar al objeto ThresholdData a un
objeto ManagedElement.
*/
          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

```
Esta vinculación de nombre se utiliza para denominar el objeto HDFile a un objeto
AbstractHDScanner.
*/
          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

```
Esta vinculación de nombre se utiliza para denominar el objeto AbstractHDScanner
a unManagedElement.
*/
          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

```
Este módulo contiene los valores constantes que determinan los formatos de
archivo de datos históricos.
*/
     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_
```

ANEXO A

Formato de fichero

El presente anexo describe el formato de fichero de datos de mediciones de la calidad de funcionamiento, denominado q822d1FileText.

A.1 BNF del formato de fichero de texto

La BNF en este anexo define el formato de fichero de datos de mediciones de la calidad de funcionamiento, denominado "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> <GranualarityPeriod>
     <Newline> <DataSetTypes> #endheader <Newline>
```

```
<MeasuredObjectType> := <String>

<GranualarityPeriod> := <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
```

NOTA 1 – El espacio en blanco contenido en el fichero no tiene ningún significado. El generador de fichero no insertará espacio en blanco en el fichero. El analizador gramatical de fichero saltará y hará caso omiso de todo espacio en blanco que se encuentre en el fichero.

NOTA 2 – Un analizador gramatical de fichero retrocompatible saltará y hará caso omiso de todos los datos entre #<tag> y #end<tag>.

NOTA 3 – Toda extensión a este formato de fichero requerirá nuevas definiciones #<tag>.

A.2 Descripción del formato de fichero de texto

Los datos históricos están contenidos en el fichero de mediciones de la calidad de funcionamiento. El fichero consiste en una definición de tipo de fichero y un conjunto de datos de mediciones. El orden de los datos es intrascendente. Deberá haber como mínimo un conjunto de datos en el fichero. En caso contrario, el fichero no se creará.

El fichero de mediciones de la calidad de funcionamiento deberá seguir las siguientes reglas de generación de fichero. (Véase la figura 7, que describe la estructura del fichero):

1) El tipo de fichero viene definido por su formato de fichero, versión de formato y paradigma de modelado. FileFormat indica el formato de fichero. FormatVersion especifica la versión de un determinado formato de fichero. El valor de FileFormat en esta Recomendación es "q822d1FileText", y el de FormatVersion es "version1".

2) El paradigma de modelado es el paradigma de modelado de sistema de gestión que genera el fichero. Si el fichero se genera de un sistema de gestión CORBA que cumple con el marco CORBA del UIT-T, el valor para ModelParadigm será "ITUTCORBA". En caso contrario,

se proporcionará un valor adecuado, por ejemplo, "CMIP", "SNMP", "TL1", "abcCORBA", etc.

3) La especificación de gestión de la calidad de funcionamiento para un determinado dominio de modelado de información deberá especificar el convenio de semántica y denominación de ModelParadigm, NodeID, MeasuredObjectType, DataSetTypeName, ParameterType, ID, MeasuredObjectIDPrefix, MeasuredObjectID.

4) MeasuredObjectIDPrefix define el prefijo común para MeasuredObjectID en una sección de nodo.

5) MeasuredObjectIDAlias define el MeasuredObjectID de alias breve para FDN MeasuredObjectID largo. El alcance de una definición de alias se extiende hasta el final del fichero a menos que haya sido redefinido por otro MeasuredObjectIDAlias.

6) DataSetTypeIndex de un DataSetValue indica el tipo de conjunto de datos correspondiente tal como está definido en el encabezamiento del cuadro.

7) Todos los datos de mediciones en un mismo cuadro tienen el mismo periodo de granularidad y representan mediciones para el mismo tipo de objeto medido.

8) Si no hay datos para un periodo determinado, el fichero no deberá contener información para dicho periodo entre #period y #endperiod, los cuales deberán estar presentes de todas formas.

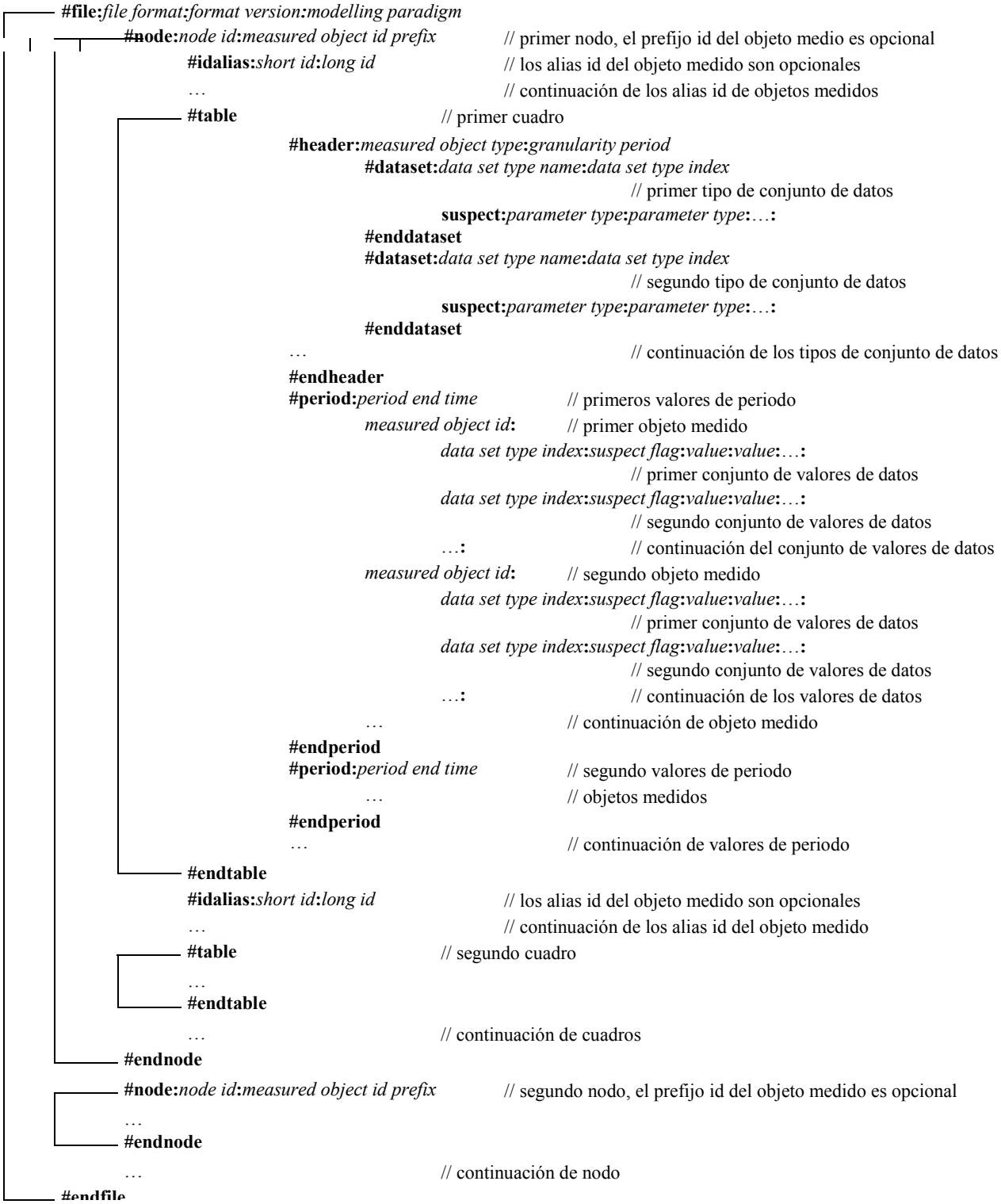

Figura 7/Q.822.1 – Ejemplo de formato de fichero PM

ANEXO B

Convenio de formato de fichero CORBA

En este anexo se describe el convenio de semántica y denominación de ModelParadigm, NodeID, MeasuredObjectType, DataSetTypeName, ParameterType, ID, MeasuredObjectIDPrefix, MeasuredObjectName en el marco CORBA del UIT-T.

1) *ModelParadigm* – será "ITUTCORBA".
2) *NodeID* – FDN de elemento gestionado, o nombre, específico de la aplicación, del elemento gestionado.
3) *MeasuredObjectType* – Nombre de interfaz, con determinación de alcance, del objeto gestionado medido.
4) *DataSetTypeName* – Nombre de interfaz, con determinación de alcance, de CurrentData o su subclase.
5) *ParameterType* – Nombre de atributo en HistoryDataValueType o su subvaluetype.
6) *ID* – FDN de nombre CORBA del objeto gestionado.
7) *MeasuredObjectIDPrefix* – RDN de nombre CORBA del objeto gestionado.
8) *MeasuredObjectID* – FDN de nombre CORBA del objeto gestionado medido.

APÉNDICE I

Ejemplo de utilización de herencia de CurrentDataValueType e HistoryDataValueType

Según se describió en 4.3.2, HistoryDataValueType se utiliza como los dispositivos de datos históricos de lectura solamente, de los datos actuales. Este valuetype será accesible mediante métodos en la interfaz CurrentData. Cuando se crea una subclase de los datos actuales se pretende que, junto con ésta, se cree una subclase del correspondiente HistoryDataValueType.

Este anexo proporciona un ejemplo de la manera de definir un HistoryDataValueType para una interfaz CurrentData dada y cómo acceder a valores de datos históricos mediante la correspondiente instancia de CurrentData.

En este ejemplo se utiliza la interfaz TrafficControlCurrentData la cual sirve para supervisar la eficacia de un control de tráfico. Esta interfaz es como sigue:

```
interface TrafficControlCurrentData: itut_q822d1::CurrentData
{
/**
el número de llamada que está actualmente afectado por el ejemplar de control de
tráfico.
*/
     short callsAffectedByTrafficControlGet ()
          raises (itut_x780::ApplicationError);

/**
el número de llamada que está actualmente afectado por el ejemplar de control de
tráfico.
*/
     short callsOfferedToTrafficControlGet ()
          raises (itut_x780::ApplicationError,
               NOcallsOfferedPackage);
}; // interface TrafficControlCurrentData
```

Sobre la base del marco NM CORBA, a fin de recibir todos los atributos soportados por un ejemplar de TrafficControlCurrentData, se definirá un tipo TrafficControlCurrentDataValueType:

```
valuetype TrafficControlCurrentDataValueType: itut_q822d1::CurrentDataValueType
{
     public short callsAffectedByTrafficControl;
          // trafficControlCurrentDataPackage
          // GET
     public short callsOfferedToTrafficControl;
          // conditional
          // callsOfferedPackage
          // GET
}; // valuetype TrafficControlCurrentDataValueType
```

Y para salvaguardar las mediciones recopiladas en un ejemplar TrafficControlCurrentData se definirá un tipo TrafficControlHistoryDataValueType:

```
valuetype TrafficControlHistoryDataValueType: itut_q822d1::HistoryDataValueType
{
     public short callsAffectedByTrafficControl;
          // GET
     public short callsOfferedToTrafficControl;
          // GET
}; // valuetype TrafficControlHistoryDataValueType
```

Dada la definición de la interfaz TrafficControlCurrentData y el TrafficControlHistoryDataValueType, se creará un ejemplar de la interfaz TrafficControlCurrentData en la fase de ejecución para supervisar la eficacia de un determinado control de tráfico a través de la interfaz TrafficControlCurrentDataFactory:

```
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
```

Al final de cada periodo de granularidad, se genera un registro de TrafficControlHistoryDataValueType y las mediciones. callsAffectedByTrafficControl y callsOfferedToTrafficControl recopiladas en el ejemplar se copiarán para pasarlas al registro de datos históricos.

Para acceder a valores datos históricos TrafficControl se invoca uno de los dos métodos: getMostRecent() y getBetween() desde el ejemplar de TrafficControlCurrentData.

Para acceder a valores de datos históricos con mayor alcance se utiliza el ejemplar de HistoryDataScanner.

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie B	Medios de expresión: definiciones, símbolos, clasificación
Serie C	Estadísticas generales de telecomunicaciones
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedios
Serie I	Red digital de servicios integrados
Serie J	Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedios
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	RGT y mantenimiento de redes: sistemas de transmisión, circuitos telefónicos, telegrafía, facsímil y circuitos arrendados internacionales
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	**Conmutación y señalización**
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	Redes de datos y comunicación entre sistemas abiertos
Serie Y	Infraestructura mundial de la información y aspectos del protocolo Internet
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación

Impreso en Suiza
Ginebra, 2002