UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T Q.834.4

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

(07/2003)

SERIE Q: CONMUTACIÓN Y SEÑALIZACIÓN

Interfaz Q3

Especificación de interfaz de arquitectura de intermediario de petición de objeto común para las redes ópticas pasivas de banda ancha basada en los requisitos de interfaz del lenguaje de modelado unificado

Recomendación UIT-T Q.834.4

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, 5, 6, R1 Y R2	Q.120–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
INTERFAZ Q3	**Q.800–Q.849**
SISTEMA DE SEÑALIZACIÓN DIGITAL DE ABONADO N.° 1	Q.850–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.834.4

Especificación de interfaz de arquitectura de intermediario de petición de objeto común para las redes ópticas pasivas de banda ancha basada en los requisitos de interfaz del lenguaje de modelado unificado

Resumen

La presente Recomendación ofrece una definición del IDL de CORBA para la interfaz de gestión entre un sistema de gestión del proveedor y un sistema de gestión del operador. En este trabajo se definen algunos aspectos de la gestión de recursos de red especificados en la serie de Recomendaciones UIT-T G.983.x relativas a los equipos de redes ópticas pasivas de banda ancha (BPON).

En términos generales, el sistema de gestión del proveedor es un sistema de gestión de elementos (EMS) mientras que el sistema de gestión del operador (OMS) es un sistema de gestión de red (NMS). Sin embargo, es preciso que el sistema de gestión del proveedor presente al sistema de gestión del operador una "perspectiva de red" de la gestión de la conexión. Por este motivo se ha estimado necesario utilizar, para mayor claridad, la terminología adoptada en la denominación de los sistemas implicados.

Por otra parte hay que tener en cuenta que la Rec. UIT-T Q.834.1 contiene un conjunto de requisitos de funcionalidad y un listado de definiciones de entidades gestionadas que constituyen la base de la información de gestión necesaria para la "perspectiva del elemento de red" de los equipos de las BPON. La Rec. UIT-T Q.834.2 completa la definición de la información de gestión de los equipos de las BPON con las definiciones de las entidades gestionadas correspondientes a la "perspectiva de red". La Rec. UIT-T Q.834.3 trata del comportamiento de la interfaz de gestión utilizando diagramas UML y descripciones de casos de empleo. En esta Recomendación y en la Rec. UIT-T Q.834.3 se hace referencia continuamente a la información de gestión modelada en las Recomendaciones UIT-T Q.834.1 y Q.834.2.

Orígenes

La Recomendación UIT-T Q.834.4 fue aprobada el 7 de julio de 2003 por la Comisión de Estudio 4 (2001-2004) del UIT-T por el procedimiento de la Recomendación UIT-T A.8. Esta edición incluye la modificaciones introducidas por el corrigendum 1 aprobado el 13 de enero de 2004.

Palabras clave

APON, BPON, CORBA, IDL, PON, UML.

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.

La observancia de esta Recomendación es voluntaria. Ahora bien, la Recomendación puede contener ciertas disposiciones obligatorias (para asegurar, por ejemplo, la aplicabilidad o la interoperabilidad), por lo que la observancia se consigue con el cumplimiento exacto y puntual de todas las disposiciones obligatorias. La obligatoriedad de un elemento preceptivo o requisito se expresa mediante las frases "tener que, haber de, hay que + infinitivo" o el verbo principal en tiempo futuro simple de mandato, en modo afirmativo o negativo. El hecho de que se utilice esta formulación no entraña que la observancia se imponga a ninguna de las partes.

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

Página

Recomendación UIT-T Q.834.4

Especificación de interfaz de arquitectura de intermediario de petición de objeto común para las redes ópticas pasivas de banda ancha basada en los requisitos de interfaz del lenguaje de modelado unificado

1 Objeto

La presente Recomendación trata del diseño de una interfaz interactiva mecanizada entre el sistema de gestión del proveedor, que gestiona los recursos de la red BPON, y un sistema de gestión del operador (OMS, *operator management system*). El diseño se basa en los diagramas UML y en las descripciones de caso de empleo de la Rec. UIT-T Q.834.3. El planteamiento general adoptado en la Rec. UIT-T Q.834.3, e implementado en ésta, consiste en que el sistema de gestión del proveedor ofrezca servicios de gestión al sistema de gestión del operador. Estos servicios proporcionan al OMS una abstracción de alto nivel de las siguientes capacidades:

- Suministro de recursos de red instalados.
- Suministro de recursos de red no instalados y entre ellos la reserva de capacidad.
- Prestación del servicio.
- Gestión de archivos.
- Gestión del soporte lógico del NE.
- Copia de seguridad y recuperación de los datos de configuración del NE.
- Gestión de la calidad de funcionamiento.
- Publicación de los eventos del NE.
- Gestión de perfiles.
- Pruebas.
- Planificación de actividades.
- Gestión de transferencias masivas.
- Sincronización NE-EMS.
- Control de acceso.

Esta Recomendación describe las interfaces IDL de CORBA que soportan los servicios enumerados *supra*.

2 Referencias

2.1 Recomendaciones relacionadas

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. En esta Recomendación, la referencia a un documento, en tanto que autónomo, no le otorga el rango de una Recomendación.

[1] Recomendación UIT-T M.3010 (2000), *Principios para una red de gestión de las telecomunicaciones*.

[2] Recomendación UIT-T M.3200 (1997), *Servicios de gestión de red de gestión de las telecomunicaciones y sectores gestionados de las telecomunicaciones: Panorama general*.

[3] Recomendación UIT-T M.3400 (2000), *Funciones de gestión de la red de gestión de las telecomunicaciones*.

[4] OMG Document formal/99-06-01, *Unified Modelling Language, Section 1*.

[5] Recomendación UIT-T G.983.1 (1998), *Sistemas de acceso óptico de banda ancha basados en redes ópticas pasivas*, más enmienda 1 (2001).

[6] Recomendación UIT-T Q.834.1 (2001), *Requisitos y entidades gestionadas de las redes ópticas pasivas basadas en el modo de transferencia asíncrono para la visión de elemento de red*.

[7] Recomendación UIT-T Q.834.2 (2001), *Requisitos y entidades gestionadas de las redes ópticas pasivas basadas en el modo de transferencia asíncrono para la visión de red*.

[8] Recomendación UIT-T Q.834.3 (2001), *Descripción del lenguaje de modelado unificado para los requisitos de interfaz de gestión de redes ópticas pasivas de banda ancha*.

[9] Recomendación UIT-T M.3020 (2000), *Metodología para la especificación de interfaces de la RGT*.

[10] Recomendación UIT-T G.983.2 (2002), *Especificación de la interfaz de control y gestión de terminales de red óptica para redes ópticas pasivas de banda ancha*.

[11] Recomendación UIT-T G.983.3 (2001), *Sistema de acceso óptico de banda ancha con capacidad de servicio incrementada mediante atribución de longitud de onda*.

[12] Recomendación UIT-T G.983.4 (2001), *Sistema de acceso óptico de banda ancha con asignación dinámica de anchura de banda para aumentar la capacidad de servicio*.

[13] Recomendación UIT-T G.983.5 (2002), *Sistema de acceso óptico de banda ancha con mayor capacidad de supervivencia*.

[14] Recomendación UIT-T G.983.6 (2002), *Especificaciones de la interfaz de gestión y control de terminales de red óptica para sistemas de red óptica pasiva de banda ancha con características de protección*.

[15] Recomendación UIT-T G.983.7 (2001), *Especificación de la interfaz de gestión y control de terminación de red óptica para sistema de red óptica pasiva de banda ancha con asignación dinámica de anchura de banda*.

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

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

2.2 Otras referencias

[18] OMG Document formal/02-09-02, *CORBA Services – Naming Service specification*.

[19] OMG Document formal/02-08-04, *CORBA Services – Notification Service specification*.

3 Términos y definiciones

3.1 Términos tomados de la Rec. UIT-T M.3010

En esta Recomendación se utiliza el siguiente término tomado de la Rec. UIT-T M.3010.

– Usuario.

3.2 Términos tomados del UML

En esta Recomendación se utilizan los siguientes términos tomados del UML [4].

– Actor.
– Clase.
– Diagrama de clase.
– Caso de empleo.

3.3 Términos tomados del servicio de denominación OMG

En esta Recomendación se utilizan los siguientes términos tomados del servicio de denominación OMG [18].

– Gráfico de denominación.
– Nombre.

3.4 Términos tomados de la Rec. UIT-T Q.834.1

En esta Recomendación se utiliza el siguiente término tomado de la Rec. UIT-T Q.834.1.

– Entidad gestionada

3.5 Términos tomados de la Rec. UIT-T Q.834.3

En esta Recomendación se utilizan los siguientes términos tomados de la Rec. UIT-T Q.834.3.

– Activar.
– Asignar.
– Autodescubrimiento.
– Recurso de la BPON.
– Construir.
– Filtrar.
– Instalar.
– Rango.
– Registro.
– Reserva.
– Ejemplar de servicio.
– Etiqueta de usuario.

3.6 Nuevos términos

En esta Recomendación se definen los términos siguientes.

3.6.1 objeto de servicio: Conjunto de objetos que facilitan el acceso a los servicios de gestión implementados en el sistema de gestión del proveedor. Las interfaces con los objetos de servicio constituyen la especificación correspondiente a la porción de IF1 definida en esta Recomendación.

3.6.2 objeto de dominio: Conjunto de objetos que definen la información de la gestión de los recursos de red de la BPON. Los objetos de dominio los suministran principalmente los listados de entidades gestionadas de las Recomendaciones UIT-T Q.834.1 y Q.834.2.

3.6.3 objeto interno: Conjunto de objetos destinados a soportar la lógica interna del sistema de gestión del proveedor. Estos objetos son completamente invisibles desde la perspectiva del OMS.

4 Abreviaturas

En esta Recomendación se utilizan las siguientes siglas.

AAL Capa de adaptación ATM (*ATM adaptation layer*)

APON ATM por la red óptica pasiva (*ATM-PON*)

ATM Modo de transferencia asíncrono (*asynchronous transfer mode*)

BICI Interfaz interoperadores de banda ancha (*broadband inter-carrier interface*)

BISSI Interfaz entre sistemas de conmutación de banda ancha (*broadband inter-switching system interface*)

BPON Red óptica pasiva de banda ancha (*broadband passive optical network*)

CAC Control de admisión de llamadas (*call admission control*)

CCITT Comité Consultivo Internacional Telegráfico y Telefónico

CES Servicio de emulación de circuitos (*circuit emulation service*)

CNM Gestión de red del cliente (*customer network management*)

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

CTP Punto de terminación de conexión (*connection termination point*)

CTT Parte de incidencias del cliente (*customer trouble ticket*)

DSx Señal digital x (*digital signal x*)

EM Gestión de elementos (*element management*)

EML Capa de gestión de elementos (*element management layer*)

EMS Sistema de gestión de elementos (*element management system*)

EOC Canal de operaciones insertado (*embedded operations channel*)

Ex Señal digital europea x (*European digital signal x*)

FSAN Red de acceso a servicios completos (*full services access network*)

GUI Interfaz de usuario gráfico (*graphical user interface*)

IDL Lenguaje de definición de interfaz (*interface definition language*)

ME Entidad gestionada (*managed entity*)

MIB Base de información de gestión (*management information base*)

NE Elemento de red (*network element*)

NMS Sistema de gestión de red (*network management system*)

NT Terminal de red (*network terminal*)

ODN Red de distribución óptica (*optical distribution network*)

OLT Terminal de línea óptica (*optical line terminal*)

OMG Grupo de gestión de objetos (*object management group*)

OMS Sistema de gestión del operador (*operator management system*)

ONT Terminal de red óptica (*optical network terminal*)

ONU Unidad de red óptica (*optical network unit*)

OS Sistema de operaciones (*operations system*)

PON Red óptica pasiva (*passive optical network*)

PVC Circuito virtual permanente (*permanent virtual circuit*)

QoS Calidad de servicio (*quality of service*)

RCD Red de comunicación de datos

RGT Red de gestión de telecomunicaciones

TCA Alerta de rebasamiento de umbral (*threshold crossing alert*)

TP Punto de terminación (*termination point*)

TTP Punto de terminación de camino (*trail termination point*)

UIT Unión Internacional de Telecomunicaciones

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

UNI Interfaz usuario-red (*user-network interface*)

VC Canal virtual (*virtual channel*)

VCC Conexión de canal virtual (*virtual channel connection*)

VCI Identificador de canal virtual (*virtual channel identifier*)

VP Trayecto virtual (*virtual path*)

VPC Conexión de trayecto virtual (*virtual path connection*)

VPI Identificador de trayecto virtual (*virtual path identifier*)

5 Convenios

En dos cláusulas de esta Recomendación se han utilizado convenios específicos. El primero es la descripción de los módulos IDL en la cláusula 8 y el segundo corresponde a los ficheros IDL que aparecen en el anexo C. Por último, en esta Recomendación se utilizan ciertos convenios que afectan a los valores nulos.

5.1 Convenio de descripción de los módulos

Las descripciones de los módulos obedecen al siguiente esquema:

– Nombre del módulo – sinopsis de los servicios, incluidos los modelos pertinentes de alto nivel, las descripciones de los elementos de datos esenciales para la empresa y los escenarios que sean precisos.

  • Nombre de la interfaz – sinopsis de los servicios específicos proporcionados por esta interfaz.

    – Nombre de la operación$_1$ – descripción detallada del comportamiento de esta operación incluida su signatura, descripción de cada parámetro de entrada y descripción del valor devuelto.

    – …

    – Nombre de la operación$_n$ – descripción detallada del comportamiento de esta operación incluida su signatura, la descripción de cada parámetro de entrada y la descripción del valor devuelto.

    – Excepciones – condiciones específicas del contexto que provocan las excepciones indicadas.

5.2 Convenio de los ficheros IDL

Los ficheros de IDL tienen el formato siguiente:

- **#ifndef __<MODULENAME>_DEFINED** (el nombre del módulo coincide con el del fichero);
- **#define __<MODULENAME>_DEFINED**
- **#include "<idlfilename>"**[1]
- **#pragma prefix "itu.Int"**
- **module q834_4 {**
- **module <modulename> {**
- todas las definiciones de tipos de datos específicos y las especificaciones de interfaz necesarias que haya indicado el diseño de la interfaz iniciado como consecuencia del examen de la Rec. UIT-T Q.834.3;
- **}; //module <modulename>**
- **}; //module q834_4**

Cada definición de módulo comienza con la definición del tipo de datos separada de la definición de la interfaz. Todos los tipos de datos definidos en esta Recomendación comienzan con un carácter alfanumérico en mayúsculas. Si los nombres de los tipos de datos están integrados por varios nombres propios, pronombres o adjetivos, cada componente de esta unión deberá comenzar con un carácter alfanumérico en mayúsculas. Las etiquetas de los parámetros utilizados en cualquier signatura de operación de interfaz comienzan con un carácter alfanumérico en minúsculas. Los nombres de las operaciones comienzan con un carácter alfanumérico en minúsculas.

5.3 Valores NULOS

En la presente Recomendación, cuando se menciona la "cadena de caracteres nula" se refiere a una cadena de caracteres vacía y no a un objeto nulo soportado por lenguajes tales como JAVA. Análogamente, la referencia a una "secuencia nula" se refiere a la transferencia de una secuencia de cero elementos.

6 Sinopsis de la arquitectura de la interfaz

La figura 1 ilustra la arquitectura de red de la BPON y su sistema de gestión. Esta tecnología proporciona un mecanismo integrado de entrega de acceso por bucle, para los servicios de telecomunicaciones que los operadores tienen desplegados actualmente. Entre éstos se encuentran los servicios de telefonía y de calidad vocal, la emulación de circuitos, Ethernet, ATM, xDSL y el vídeo. Los trabajos de normalización en curso siguen estudiando soluciones de transporte para las capas 1 y 2 de la ODN, aunque ya existen implementaciones en algunas redes de operadores con APON y ATM, con arreglo a las definiciones de la Rec. UIT-T G.983.1.

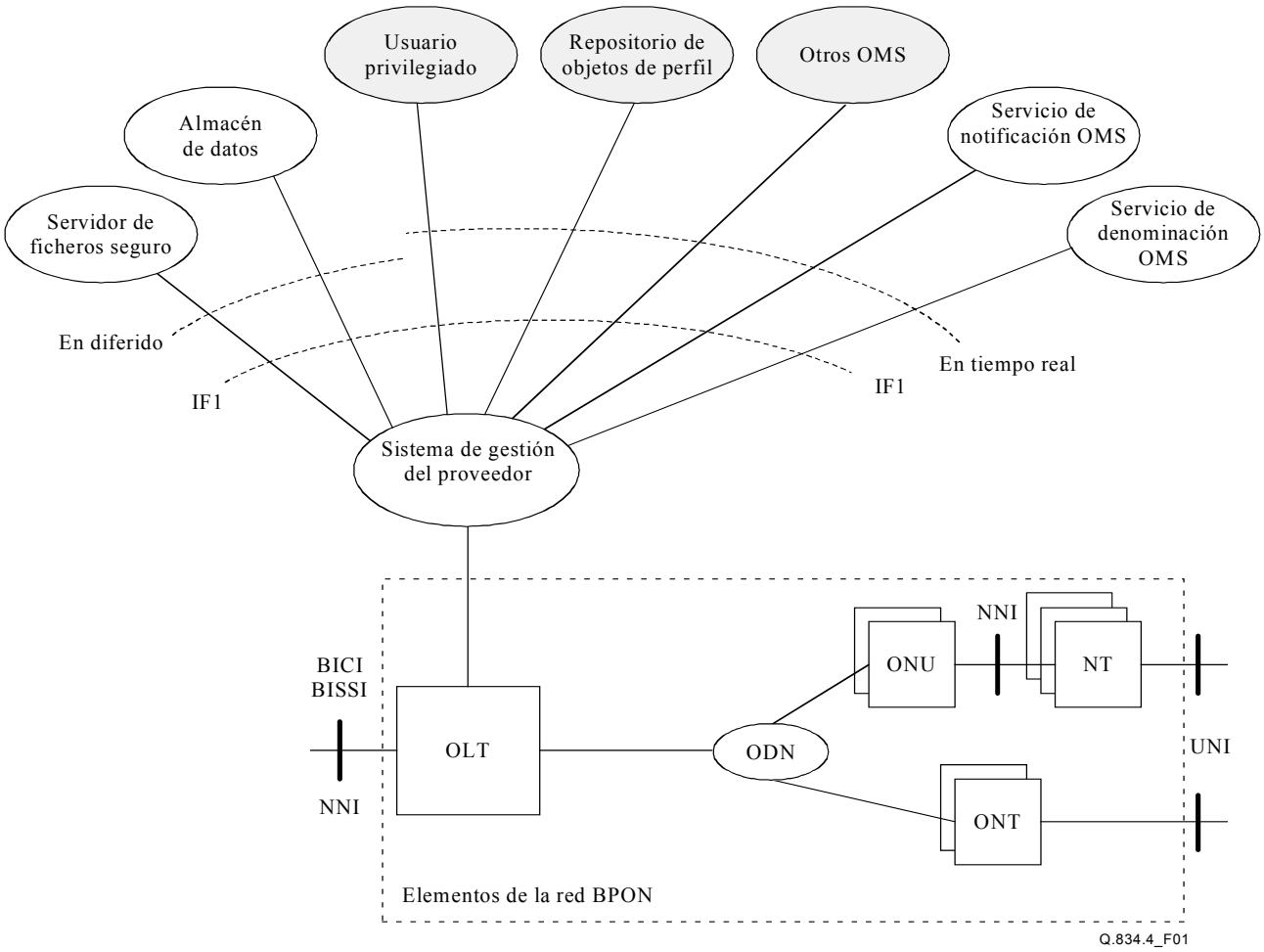
En la figura se muestra asimismo la interfaz IF1 (Q) entre el sistema de gestión del proveedor y los sistemas de gestión del operador. La interfaz IF1 cubre todos los aspectos de la gestión de suministro de la red y de prestación del servicio, la gestión de la calidad de funcionamiento de la red, la gestión del tráfico, el mantenimiento, las pruebas y la administración de la seguridad de los usuarios. La especificación de IF1 habría constituido un problema muy difícil de resolver de no haber adoptado el planteamiento de buscar un nivel de abstracción por encima de los detalles de la tecnología de entrega del transporte y de las complejidades propias de la gestión de tantos tipos de servicios. Forman parte de este planteamiento la definición y descripción de los requisitos de

---

[1] El fichero Q834Common.idl file se incluye junto con uno de cada dos ficheros IDL en esta Recomendación.

interfaz mediante el lenguaje de modelado unificado, se ha aplicado la metodología de la Rec. UIT-T M.3020 y se ha documentado en la Rec. UIT-T Q.834.3.

Los sistemas de operador identificados en este diagrama corresponden directamente a los actores de sistemas definidos en la Rec. UIT-T Q.834.3. Esta Recomendación especifica las transacciones interactivas en tiempo real que afectan al sistema de gestión del proveedor, aunque no describe explícitamente las estructuras de los ficheros transferidos entre el sistema de gestión del proveedor y el almacén de datos o el servidor de ficheros seguro. Al estar especificadas las interfaces con el servicio de denominación OMG y con el servicio de notificación OMG, el objetivo de esta Recomendación se centra en los mensajes entre el sistema de gestión del proveedor y los actores de sistemas sombreados en la figura.



BICI	Interfaz interoperadores de banda ancha	ODN	Red de distribución óptica
BISSI	Interfaz entre sistemas de conmutación de banda ancha	OLT	Terminal de línea óptica
BPON	Red óptica pasiva de banda ancha	ONU	Unidad de red óptica
NNI	Interfaz red-red	ONT	Terminal de red óptica
NT	Terminal de red	UNI	Interfaz usuario-red

Figura 1/Q.834.4 – Sinopsis de la arquitectura de la interfaz

7 Nombres y restricciones a su aplicación

Esta cláusula ofrece directrices para los nombres de objetos y los identificadores de entidades gestionadas referenciadas en IF1.

Al analizar los diagramas de clase desarrollados en la Rec. UIT-T Q.834.3 se observa que el sistema de gestión del proveedor gestiona tres tipos diferentes de "objetos" que pueden clasificarse del siguiente modo:

- Objetos de servicio: proporcionan acceso a los servicios de gestión implementados en el sistema de gestión del proveedor. Las interfaces de estos servicios son el principal objetivo de la presente Recomendación. Como ejemplos se pueden citar NERegistrar, ServiceProvisioner, TestActionPerformer y DownloadMgr. Se alinean con la Rec. UIT-T X.780 como derivaciones de la interfaz de objeto gestionado de esta Recomendación.
- Objetos de dominio: definen la información de gestión de los recursos de la red BPON. Los objetos de dominio son principalmente recursos (entidades gestionadas) definidos en las Recomendaciones UIT-T Q.834.1 y Q.834.2. Estos objetos de dominio, y los definidos en la Rec. UIT-T M.3120 y en otras Recomendaciones que se ajustan al marco CORBA del UIT-T, pueden estar disponibles para su utilización en IF1. En esta misma cláusula se presenta un mecanismo de armonización de los objetos de servicio y de los objetos de dominio.
- Objetos internos: se utilizan para soportar la lógica interna del sistema de gestión del proveedor. Son completamente invisibles para la perspectiva del OMS.

7.1 Objetos de servicio y servicio de denominación OMG

Todos los objetos de servicio (ya sean ejemplificados por el sistema de gestión del proveedor o por el OMS) tienen su interfaz IDL definida en la presente Recomendación y sus referencias de objetos registradas en el servicio de denominación OMG representado en la figura 1 utilizando la operación bind() de la interfaz del servicio de denominación. Por consiguiente, la posición del ejemplar de objeto puede determinarse por medio de la operación resolve() del servicio de denominación, cuando sea necesario.

Los nombres de objeto pueden ser "conocidos" (lo que significa que estos nombres se han documentado y acordado antes del tiempo de ejecución) o comunicados en tiempo de ejecución mediante otra interfaz. En esta Recomendación todos los objetos de servicio son "conocidos". El nombre de un objeto conocido puede documentarse mediante un gráfico de denominación. La figura 2 es un ejemplo de gráfico de denominación correspondiente a algunos de los objetos de servicio de la presente Recomendación, y utiliza el convenio de denominación descrito en la Rec. UIT-T Q.816. En el diagrama puede observarse el Id seguido de un punto y del valor de género. Sólo se representan cuatro objetos de servicio para mayor simplicidad de la representación. Los demás objetos de servicio se denominarían de una manera similar.

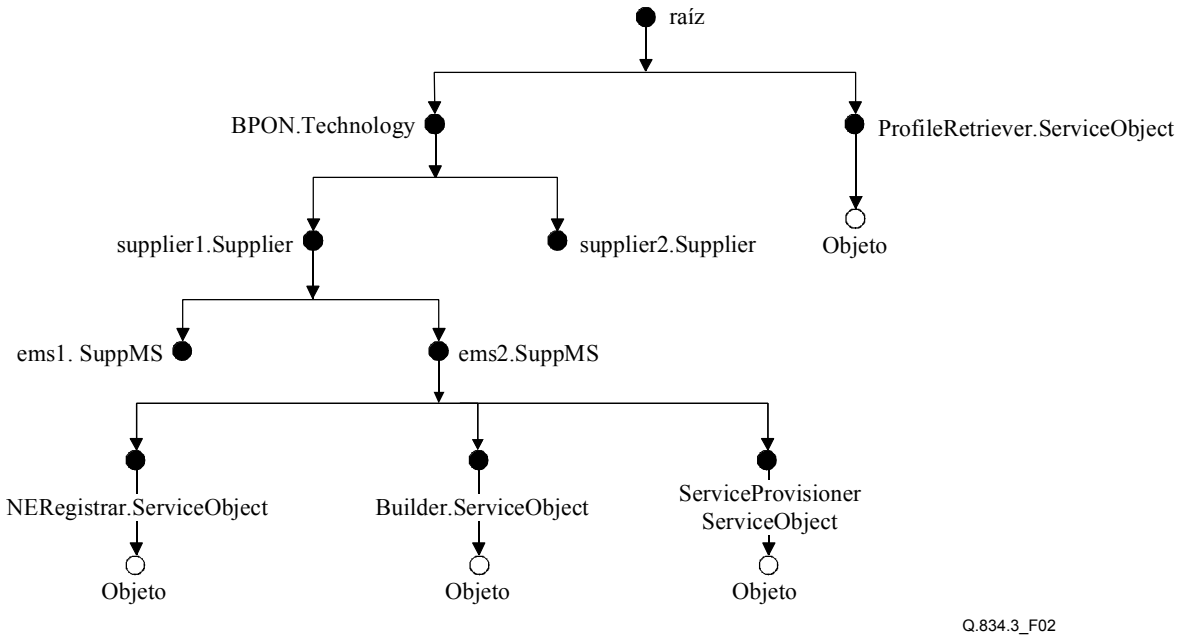


Figura 2/Q.834.4 – Gráfico de denominación

Para un objeto "conocido" es necesario el nombre completo de su ruta de acceso. Cuando se arranca un sistema de gestión del proveedor de nueva instalación, todos los ejemplares de objetos de servicio se registran automáticamente en un único servicio de denominación OMG mantenido por el operador. Los nombres de objeto registrados en el servicio de denominación OMG obedecen a la sintaxis siguiente:

```
typedef     string      Istring;

struct NameComponent {
       Istring id;
       Istring kind;
};

typedef sequence<NameComponent> Name;
```

Una interpretación del gráfico de denominación de la figura 2 es que el valor del campo "género" de todos los componentes de denominación de los objetos de servicio es la cadena de caracteres nula y que cada uno de los nodos mostrados representa los valores del campo "id" . Los detalles del gráfico de denominación así como su interpretación deben determinarse mediante acuerdo entre proveedor y operador, no siendo dichos detalles objeto de la presente Recomendación.

7.2 Objetos de dominio

El sistema de gestión del proveedor mantiene un gran número de entidades gestionadas (u objetos de dominio)[2]. Estas entidades se gestionan indirectamente a través de los servicios de gestión ofrecidos por el sistema de gestión del proveedor. No obstante, se suelen necesitar referencias (identificadores) de ejemplares específicos de recursos para que el OMS pueda utilizar de forma eficaz los servicios de gestión del sistema de gestión del proveedor. Cuando se devuelva el

[2] Si el sistema de gestión del proveedor gestiona entre 100 y 200 sistemas BPON (requisito de tamaño conveniente) y soporta totalmente la base de conocimiento de gestión compartida a la que se refieren las Recomendaciones UIT-T Q.843.1 y Q.834.2, el número de ejemplares de entidades gestionadas soportado por el sistema de gestión del proveedor varía entre decenas y miles de millones, dependiendo de la oferta de servicios.

identificador como resultado de una operación de gestión (o se incluya en las notificaciones) se supondrá que el identificador será "conocido" en las posteriores invocaciones del OMS al servicio de gestión. En la presente Recomendación, se ha proporcionado al identificador de la entidad gestionada una sintaxis que permite al sistema de gestión del proveedor informar al OMS de una de las tres maneras posible, como mínimo, de interacción con la entidad gestionada en las invocaciones posteriores. Si la entidad gestionada es un objeto gestionado disponible en IF1, el nombre CORBA del objeto se proporciona con arreglo a la descripción de la Rec. UIT-T X.780. Si la entidad gestionada tiene un objeto fachada de soporte, disponible en IF1, el nombre CORBA del objeto fachada se proporciona con arreglo a lo descrito en la Rec. UIT-T X.780.1. Si aún no estuviera disponibles ninguna de estas capacidades, el identificador consistiría en una referencia única en el contexto del dominio de gestión del sistema de gestión del proveedor. En general, los detalles de esta referencia se basan en una jerarquía específica de contención determinada mediante acuerdo entre proveedor y operador, cuyos detalles escapan al propósito de la presente Recomendación.

8 Organización de los ficheros IDL

La especificación de interfaz propuesta que aparece en esta Recomendación consta de un módulo progenitor denominado "q834_4". El módulo progenitor se divide en ficheros IDL (o sea en módulos más pequeños) correspondientes a módulos más pequeños con nombre. Cada uno de estos ficheros contiene una o más definiciones de interfaz que representan un servicio de gestión específico soportado en IF1. El cuadro 1 muestra un listado que relaciona el número de referencia del anexo C, el nombre del fichero IDL, las interfaces IDL contenidas y la referencia a la descripción de caso de empleo asociado de la Rec. UIT-T Q.834.3.

Cuadro 1/Q.834.4 – Organización de los módulos q834_4

Referencia del anexo	Ficheros IDL	Interfaz IDL	Título del caso de empleo
C.1	Q834AccessControl	AccessControlMgr	Control de acceso
C.2	Q834Build	Builder	Construcción de recursos de la BPON
C.3	Q834Common	ProbableCause MonitoringParameter RecordSetType PMCategory PhysicalLayerLoopback	
C.4	Q834ControlArchive	RecordSetMgr	Archivo de control
C.5	Q834SoftwareDownload	DownloadMgr	Soporte lógico distribuido
		VersionRepository	Control de versiones del soporte lógico del NE
C.6	Q834EventPublisher	AlarmEventSupplier SecurityEventSupplier DiscoveryEventSupplier	Publicar eventos
C.7	Q834MIBTransfer	MIBMover	Recuperación del NE

Cuadro 1/Q.834.4 – Organización de los módulos q834_4

Referencia del anexo	Ficheros IDL	Interfaz IDL	Título del caso de empleo
C.8	Q834PerformanceManager	ImpairmentPersistence	RCAA y RCIA
		ReportController	Control de informes de supervisión de la calidad de funcionamiento y del tráfico
C.9	Q834ProfileManager	ProfileConsumer ProfileUsageMgr ProfileRetriever	Gestión de objetos de perfil
C.10	Q834Registrar	NERegistrar	Suministro de recursos BPON instalados
C.11	Q834ResourceAllocation	ResourceAllocator	Recursos de reserva
C.12	Q834SchedulerManagement	SchedulerMgr	Planificador
C.13	Q834ServiceProvisioning	ServiceProvisioner	Prestación del servicio
C.14	Q834Synchroniser	Synchroniser	Proporcionar listados resumen de eventos actuales
			Sincronización del NE
C.15	Q834Test	TestActionPerformer	Pruebas
C.16	Q834FileTransfer	TransferMgr	Transferencia masiva

9 Módulos

9.1 Módulo AccessControl (*control de acceso*)

Este módulo describe la funcionalidad destinada a crear, suprimir, asignar y utilizar la información del control de acceso para los operadores que utilizan la aplicación cliente GUI del sistema de gestión del proveedor. En este módulo los usuarios pueden asignarse a grupos de usuarios. Pueden concederse permisos a usuarios individuales o a grupos de usuarios. El usuario posee los permisos de cualquier grupo al que haya sido asignado. Cualquier permiso definido individualmente para un usuario específico (véanse 9.1.1.14 y 9.1.1.15) tiene prioridad sobre una asignación de grupo. En las asignaciones de grupo el máximo nivel de permiso tiene prioridad para cualquier usuario asignado a los grupos. Se supone que el usuario privilegiado no solicita actividades objetivo ambiguas durante la creación o modificación de un grupo de usuarios ni durante la especificación del permiso correspondiente a un usuario individual.

Una actividad objetivo es definida según se muestra en el cuadro 2 en todo el módulo:

Cuadro 2/Q.834.4 – Detalle de la actividad objetivo

Nombre del campo	Definición	Sintaxis	Observaciones
activityLevel	Especifica el nivel de acceso de la actividad	enum	monitorOnly allowedToExecute noAccess
activityType	Especifica el tipo de actividad	short	Definido como varias constantes en la interfaz AccessControlMgr
administrationDomainList	Identificador proporcionado por el OMS o por el operador durante el registro para indicar el dominio de administración al que pertenece el NE	UserLabel	

9.1.1 Interfaz AccessControlMgr (*gestor del control de accesos*)

9.1.1.1 setPasswordPolicy (*establecer política de contraseñas*)

Esta operación permite a los usuarios privilegiados gestionar la política de contraseñas.

A continuación se muestra la signatura de la operación **setPasswordPolicy**:

```
void setPasswordPolicy (in PasswordPolicyType passwordPolicy)
                        raises (AccessDenied);
```

El parámetro de entrada **passwordPolicy** identifica la política de contraseñas aplicada por el sistema de gestión del proveedor y consta de UserLoginPolicy (*política de entrada en comunicación de usuario*) y SessionPolicy (*política de sesión*). UserLoginPolicy dicta los parámetros siguientes relativos al Id de usuario y contraseña: tamaño mínimo del Id de usuario, tamaño mínimo de la contraseña del usuario, plazo en días antes de poder utilizar de nuevo esta contraseña, número máximo de intentos de entrada en comunicación permitidos antes de bloquear el acceso del usuario durante un día, validez (en días) de la contraseña, obligatoriedad de la composición alfanumérica de la contraseña, obligatoriedad de la inclusión de algún carácter especial, posibilidad de que la contraseña contenga caracteres repetidos y posibilidad de que el Id de usuario forme parte de la contraseña.

SessionPolicy identifica los siguientes parámetros relativos a la sesión: plazo de inactividad de una sesión antes de su cancelación, plazo de inactividad de un Id de usuario para su desactivación y máximo número de sesiones activas permitidas por Id de usuario.

Como en un sistema de gestión del proveedor sólo puede haber una passwordPolicy, no es necesario crear un método ya que o bien existe una política por defecto en el sistema de gestión del proveedor o el primer método establecido puede actuar como set-by-create (definidor en creación).

El valor devuelto es del tipo **void**.

9.1.1.2 passwordPolicyGet (*obtener política de contraseñas*)

Esta operación permite a los usuarios privilegiados recuperar la política relativa a la sintaxis de los datos intercambiados y de los temporizadores utilizados cuando se entra en conexión con el sistema de gestión del proveedor.

A continuación se muestra la signatura de la operación **passwordPolicyGet**:

```
PasswordPolicyType passwordPolicyGet()
                   raises (AccessDenied);
```

Esta operación no necesita parámetros de entrada.

El valor devuelto es del tipo **PasswordPolicyType** (*tipo de política de contraseñas*) y proporciona los detalles que rigen la entrada en sesión del usuario con el sistema de gestión del proveedor.

9.1.1.3 userListGet (*obtener lista de usuarios*)

Esta operación permite a los usuarios privilegiados recuperar la lista de identificadores de usuario con algún tipo de acceso al sistema de gestión del proveedor, así como sus actividades objetivo y los grupos a los que pertenecen.

A continuación se muestra la signatura de la operación **userListGet**:

```
UserSeqType userListGet ()
                   raises (AccessDenied);
```

Esta operación no necesita parámetros de entrada.

El valor devuelto es del tipo **UserSeqType** y proporciona la lista de identificadores de usuario que tienen algún tipo de acceso al sistema de gestión del proveedor así como sus actividades objetivo y los grupos a los que pertenecen.

9.1.1.4 userGroupListGet (*obtener lista de grupos de usuarios*)

Esta operación permite a los usuarios privilegiados recuperar los grupos de usuarios con acceso al sistema de gestión del proveedor, con la identificación de los miembros de cada grupo y de sus actividades objetivo.

A continuación se muestra la signatura de la operación **userGroupListGet**:

```
UserGroupSeqType userGroupListGet ()
                   raises (AccessDenied);
```

Esta operación no necesita parámetros de entrada.

El valor devuelto es del tipo **UserGroupSeqType** y proporciona los grupos de usuarios con acceso al sistema de gestión del proveedor y la identificación de los miembros de los grupos y de sus actividades objetivo.

9.1.1.5 userGet (*obtener usuarios*)

Esta operación permite a los usuarios privilegiados recuperar la filiación del grupo del usuario y sus actividades objetivo.

A continuación se muestra la signatura de la operación **userGet**:

```
UserType userGet (
          in UserIdType userId )
          raises (AccessDenied, UnknownUserIds);
```

El parámetro de entrada **userId** identifica al usuario de interés.

El valor devuelto es del tipo **UserType** y proporciona la filiación del grupo del usuario y sus actividades objetivo.

9.1.1.6 userGroupGet (obtener grupo del usuario)

Esta operación permite a los usuarios privilegiados recuperar a los usuarios pertenecientes al grupo y sus actividades objetivo permitidas.

A continuación se muestra la signatura de la operación **userGroupGet**:

```
UserGroupType userGroupGet (
                    in UserLabelType userGroupId)
                    raises (AccessDenied, UnknownUserGroupId);
```

El parámetro de entrada **userGroupId** identifica el grupo del usuario.

El valor devuelto es del tipo **UserGroupType** y proporciona el identificador del grupo del usuario, los usuarios pertenecientes al grupo y sus actividades objetivo permitidas.

9.1.1.7 createUserGroup (*crear grupo de usuarios*)

Esta operación permite a los usuarios privilegiados crear un nuevo grupo de usuarios. El grupo de usuarios se utiliza para otorgar a algunos usuarios ciertos accesos a determinados elementos de la red. El usuario privilegiado crea el nuevo grupo proporcionando la lista de actividades permitida al grupo de usuarios.

A continuación se muestra la signatura de la operación **createUserGroup**:

```
void createUserGroup (
                    in UserLabelType userGroupId,
                    in TargetActivitySeqType targetAdditions)
                    raises (DuplicateUserGroupId, UnknownTargets, AccessDenied);
```

El parámetro de entrada **userGroupId** identifica el grupo que ha de crearse. No se permite que el valor de UserGroupId sea la cadena de caracteres vacía. El parámetro de entrada **targetAdditions** identifica las actividades objetivo permitidas a los usuarios pertenecientes a este grupo.

El valor devuelto es del tipo **void**.

9.1.1.8 modifyUserGroup (*modificar grupo de usuarios*)

Esta operación permite a los usuarios privilegiados añadir y suprimir actividades objetivo del grupo de usuarios. El usuario privilegiado especifica el grupo de usuarios y la lista de actividades objetivo a añadir o suprimir. El sistema de gestión del proveedor procesa las supresiones antes que las adiciones. Esto permite, por ejemplo, incrementar el nivel de permiso de una determinada actividad.

A continuación se muestra la signatura de la operación **modifyUserGroup**:

```
TargetActivitySeqType modifyUserGroup (
                        in UserLabelType userGroupId,
                        in TargetActivitySeqType targetAdditions,
                        in TargetActivitySeqType targetDeletions)
                        raises (UnknownUserGroupId, UnknownTargets,
                        AccessDenied);
```

El parámetro de entrada **userGroupId** identifica el grupo de usuarios en el que el operador desea añadir o suprimir actividades objetivo. El parámetro de entrada **targetAdditions** identifica las actividades objetivo que es preciso añadir al grupo. El parámetro de entrada **targetDeletions** identifica las actividades objetivo que es preciso suprimir del grupo.

El valor devuelto es del tipo **TargetActivitySeqType** y proporciona la lista actualizada de las actividades objetivo permitidas a los usuarios que pertenecen a este grupo de usuarios.

9.1.1.9 deleteUserGroup (*suprimir grupo de usuarios*)

Esta operación permite a los usuarios privilegiados suprimir un grupo de usuarios existente. La supresión sólo puede llevarse a cabo si el grupo de usuarios está vacío.

A continuación se muestra la signatura de la operación **deleteUserGroup**:

```
void deleteUserGroup (
              in UserLabelType userGroupId)
              raises (AccessDenied, UserGroupNotEmpty, UnknownUserGroupId);
```

El parámetro de entrada **userGroupId** identifica el grupo a suprimir.

El valor devuelto es del tipo **void**.

9.1.1.10 addUsersToGroup (*añadir usuarios a un grupo*)

Esta operación permite a los usuarios privilegiados añadir nuevos usuarios a un grupo de usuarios existente.

A continuación se muestra la signatura de la operación **addUsersToGroup**:

```
void addUsersToGroup (
              in UserLabelType userGroupId,
              in UserIdSeqType userIdList)
              raises (AccessDenied, UnknownUserGroupId);
```

El parámetro de entrada **userGroupId** identifica el grupo al que es preciso añadir nuevos usuarios. El parámetro de entrada **userIdList** identifica el conjunto de identificadores de usuario que es preciso añadir al grupo de usuarios existente.

El valor devuelto es del tipo **void**.

9.1.1.11 deleteUsersFromGroup (*suprimir usuarios de un grupo*)

Esta operación permite a los usuarios privilegiados suprimir usuarios de un grupo.

A continuación se muestra la signatura de la operación **deleteUsersFromGroup**:

```
void deleteUsersFromGroup (
              in UserLabelType userGroupId,
              in UserIdSeqType userIdList)
              raises (AccessDenied, UnknownUserGroupId, UnknownUserIds );
```

El parámetro de entrada **userGroupId** identifica el grupo del que es preciso suprimir usuarios. El parámetro de entrada **userIdList** identifica el conjunto de identificadores de usuario que hay que suprimir del grupo.

El valor devuelto es del tipo **void**.

9.1.1.12 getPermissionList (*obtener lista de permisos*)

Esta operación permite a los usuarios privilegiados obtener la lista de actividades permitidas a un usuario específico.

A continuación se muestra la signatura de la operación **getPermissionList**:

```
TargetActivitySeqType getPermissionList (
                        in UserIdType userId)
                        raises (UnknownUserIds, AccessDenied);
```

El parámetro de entrada **userId** identifica al usuario cuyas actividades objetivo pretende obtener el usuario privilegiado.

El valor devuelto es del tipo **TargetActivitySeqType** y proporciona la lista de actividades objetivo permitidas a un usuario específico.

9.1.1.13 modifyPermissionList (*modificar lista de permisos*)

Esta operación permite a los usuarios privilegiados modificar la lista de actividades permitidas a un usuario. El usuario privilegiado especifica el Id de usuario y la lista de actividades que es preciso

añadir o suprimir. El sistema de gestión del proveedor procesa las supresiones antes que las adiciones. Esto permite, por ejemplo, incrementar el nivel de permisos de una determinada actividad.

A continuación se muestra la signatura de la operación **modifyPermissionList**:

```
TargetActivitySeqType modifyPermissionList (
                        in UserIdType userId,
                        in TargetActivitySeqType targetAdditions,
                        in TargetActivitySeqType targetDeletions)
                        raises (UnknownUserIds, UnknownTargets, AccessDenied);
```

El parámetro de entrada **userId** identifica al usuario cuyas actividades permitidas desea modificar el usuario privilegiado. El parámetro de entrada **targetAdditions** identifica las actividades permitidas que hay que añadir a un usuario específico. El parámetro de entrada **targetDeletions** identifica las actividades permitidas que hay que suprimir de un usuario específico.

Si un usuario pertenece a varios grupos de la misma actividad, el usuario asume el nivel de acceso más alto. Si el usuario y el grupo al que pertenece se añaden al mismo dominio de administración, el nivel de actividad del usuario tiene prioridad sobre el nivel de actividad del grupo.

El valor devuelto del tipo **TargetActivitySeqType** proporciona la lista de actividades permitidas a un usuario específico tras la modificación.

9.1.1.14 createUser (*crear usuario*)

Esta operación permite a los usuarios privilegiados crear un nuevo usuario. El usuario privilegiado proporciona un nuevo Id de usuario, su contraseña y la lista de actividades permitidas al usuario.

A continuación se muestra la signatura de la operación **createUser**:

```
void createUser (
            in UserIdType userId,
            in PasswordType password,
            in TargetActivitySeqType targetAdditions)
            raises (DuplicateUserId, UnknownTargets, AccessDenied,
            UserLoginPolicyViolation);
```

El parámetro de entrada **userId** determina la información de identificación que ha de asociarse al nuevo usuario. El parámetro de entrada **password** identifica la contraseña que ha de asociarse al nuevo Id de usuario. El parámetro de entrada **targetAdditions** identifica las actividades objetivo permitidas al nuevo usuario.

El valor devuelto es del tipo **void**.

9.1.1.15 deleteUser (*suprimir usuario*)

Esta operación permite a los usuarios privilegiados suprimir un usuario existente.

A continuación se muestra la signatura de la operación **deleteUser**:

```
void deleteUser (
            in UserIdType userId)
            raises (UnknownUserIds, AccessDenied);
```

El parámetro de entrada **userId** identifica el usuario a suprimir.

El valor devuelto es del tipo **void**.

9.1.1.16 resetPassword (*renovar contraseña*)

Esta operación permite a los usuarios privilegiados renovar la contraseña de un usuario. Esta operación tiene lugar cuando la contraseña antigua ya no está disponible y el usuario privilegiado

necesita una nueva contraseña para el usuario. Cuando se utilice por primera vez, el sistema de gestión del proveedor invitará al usuario a modificar su contraseña.

A continuación se muestra la signatura de la operación **resetPassword**:

```
void resetPassword (
              in UserIdType userId,
              in PasswordType newPassword)
              raises (UnknownUserIds, UserLoginPolicyViolation, AccessDenied);
```

El parámetro de entrada **userId** identifica el usuario que es preciso suprimir. El parámetro de entrada **newPassword** especifica la contraseña que es preciso asociar al Id de usuario.

El valor devuelto es del tipo **void**.

9.1.1.17 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema no tiene autorización de acceso al objeto de interfaz.

La excepción **DuplicateUserId** (**Id de *usuario duplicado***) se plantea cuando el Id de usuario especificado para la creación de un nuevo usuario ya existe en la base de datos del sistema de gestión del proveedor.

La excepción **DuplicateUserGroupId** (**Id de *grupo de usuario duplicado***) se plantea cuando el Id de grupo de usuario especificado para la creación de un nuevo grupo de usuarios ya existe en el sistema de gestión del proveedor. Es decir, el sistema de gestión del proveedor supervisa la asignación de etiquetas de usuario a los identificadores de grupo de usuarios, para todos los usuarios reconocidos por éste.

La excepción **UnknownTargets** (***objetivos desconocidos***) se plantea cuando no puede identificarse la lista especificada de actividades objetivo.

La excepción **UnknownUserGroupId** (**Id de *grupo de usuarios desconocido***) se plantea cuando el Id de grupo de usuarios especificado no puede identificarse.

La excepción **UnknownUserIds** (**Id de *usuarios desconocidos***) se plantea cuando no se reconoce algún Id de usuario.

La excepción **UserGroupNotEmpty** (***grupo de usuarios no vacío***) se plantea cuando hay un intento de suprimir un grupo de usuarios que no está vacío (es decir, hay usuarios registrados en el grupo).

La excepción **UserLoginPolicyViolation** (***violación de la política de entrada en sesión de usuarios***) se plantea cuando la contraseña especificada para un nuevo usuario o para modificar la contraseña viola la política de entrada en comunicación de los usuarios.

9.2 Construir módulo

El sistema de gestión del proveedor construye grupos de modelos de gestión para equipos planificados a petición de un OMS o de un operador como parte de las actividades previas al suministro. Estos recursos pueden ser nodos (OLT, ONT, ONU, NT) o unidades enchufables (*plug-in*). Si hay equipos instalados, se utilizan operaciones de modificación para completar el suministro de los recursos instalados. Se establecen grupos de protección o bien para los recursos previos al suministro o para los recursos instalados.

9.2.1 Interfaz constructora

9.2.1.1 buildNode (*construir nodo*)

Esta operación construye un NE en el sistema de gestión del proveedor. Como resultado de esta operación se crea automáticamente un conjunto de entidades gestionadas en el modelo de

información de gestión mantenido por el sistema de gestión del proveedor. Dependiendo de la implementación de los equipos del proveedor, pueden crearse, entre otros, ejemplares de equipmentHolderF para estantes y ranuras, ejemplares de NEFSAN y ejemplares de physicalPathTPF para puertos integrados.

Si el NE es un ONT o una ONU, esta actividad suprime automáticamente de la anchura de banda disponible en el sistema, la necesaria para soportar el canal de operaciones integradas entre el OLT y el ONT o la ONU.

A continuación se muestra la signatura de la operación **buildNode**:

```
ManagedEntityIdType buildNode (
                    in NEKindType nEKind,
                    in string supplierName,
                    in string location,
                    in VersionType version,
                    in SerialNumType serialNum,
                    in NameSeqType alarmSeverityProfiles,
                    in NameSeqType thresholdDataProfiles,
                    in SlotAssignmentSeqType slotAssignmentList,
                    in ManagedEntityIdType port,
                    in string modelCode,
                    in string systemTitle,
                    in VersionSeqType softwareVersions,
                    in UserLabelType nEUserLabel,
                    in ExternalTimeType externalTime,
                    in SystemTimingType systemTiming,
                    in AdministrationDomainType administrationDomain)
                    raises (UnrecognisedVersion, InvalidSerialNumSyntax,
                    DuplicateSerialNumber, UnknownProfiles,
                    UnknownManagedEntity, DuplicateUserLabel, AccessDenied,
                    InvalidExternalTime, UnknownSystemTimingSource,
                    ProfileSuspended);
```

El parámetro de entrada **nEKind** identifica el tipo de NE a construir. Las posibles opciones de tipo de NE son: OLT, ONT, ONU o NT. Las entradas **supplierName** y **version** identifican al fabricante del NE y la versión de los dispositivos físicos del NE a construir. El parámetro de entrada **location** designa la ubicación física del NE planificado. La entrada **serialNum** proporciona una cadena de caracteres única correspondiente al NE. La entrada **alarmSeverityProfiles** identifica los perfiles para la configuración de la gravedad de las alarmas individuales que ha de comunicar el NE. La entrada **thresholdDataProfiles** identifica los perfiles a utilizar cuando se configuran valores umbral para generar las TCA del NE. La entrada **slotAssignmentList** identifica la asignación de unidades enchufables a las ranuras del NE. Si un número de ranura no figura en la lista, se supone que no existe asignación para la misma y que no hay, por tanto restricciones sobre el tipo de tarjeta que puede insertarse en la ranura. Lo mismo ocurre cuando el valor de plugIn es la cadena de caracteres vacía. La entrada **port** identifica el puerto PON del OLT si el NE que se construye es un ONT o una ONU. Cuando se construye un NT, la entrada **port** identifica el puerto de la ONU que atiende al NT. Cuando se construye un OLT, el valor de **port** es nulo. La entrada **modelCode** proporciona una cadena de caracteres única que identifica el tipo de recurso de red. Este parámetro de entrada es interesante sobre todo cuando se suministra previamente un ONT o un NT. La entrada **systemTitle** identifica la etiqueta definida por el operador que ha de aplicarse al nodo. La entrada **softwareVersions** identifica las versiones del soporte lógico que debe utilizar el nodo. La entrada **nEUserLabel** proporciona una denominación de operador única para el NE construido. La entrada **externalTime** establece la hora actual expresada como hora generalizada para el NE. La entrada **systemTiming** identifica la fuente de reloj de entrada del NE que se utilizará para la sincronización de la temporización. Cuando se pase no especificado para el tipo de datos "enum" **systemTiming** desde el OMS, el NE utilizará su fuente de temporización por defecto. La entrada **administrationDomain** identifica el dominio al que pertenece el NE.

El valor devuelto es del tipo **ManagedEntityIdType** y proporciona un identificador para el nuevo NE creado por esta operación.

9.2.1.2 assignUserLabelsToNE (*asignar etiquetas de usuario al NE*)

Esta operación asigna denominaciones administrativas del operador a los NE. Esta operación es necesaria cuando el OMS se entera por primera vez de la existencia del NE por autodescubrimiento.

A continuación se muestra la signatura de la operación **assignUserLabelsToNE**:

```
void assignUserLabelsToNE (
              in SerialNumType serialNum,
              in UserLabelType nEUserLabel,
              in AdministrationDomainType administrationDomain)
              raises (InvalidSerialNumSyntax, DuplicateSerialNumber,
              DuplicateUserLabel, AccessDenied);
```

La entrada **serialNum** identifica el NE específico. La entrada **nEUserLabel** proporciona un nombre asignado por el operador al NE. La entrada **administrationDomain** identifica el dominio al que se asigna el NE.

El valor devuelto es del tipo **void**.

9.2.1.3 modifyNode (*modificar nodo*)

Esta operación inicia la reconfiguración y actualización de parámetros específicos asociados al NE.

A continuación se muestra la signatura de la operación **modifyNode**:

```
void modifyNode (
            in ManagedEntityIdType managedEntityId,
            in SlotAssignmentSeqType newSlotAssignmentList,
            in NameSeqType newAlarmSeverityProfiles,
            in NameSeqType newThresholdDataProfiles,
            in ManagedEntityIdType port,
            in string newModelCode,
            in UserLabelType newNEUserLabel,
            in ExternalTimeType externalTime,
            in AdministrationDomainType administrationDomain)
            raises(UnknownManagedEntity, UnknownNE, InvalidSlotAssignmentList,
            UnknownProfiles, DuplicateUserLabel, AccessDenied,
            InvalidExternalTime, ProfileSuspended );
```

La entrada **managedEntityId** identifica el NE objetivo a modificar. La **newSlotAssignmentList** identifica una nueva asignación de unidades enchufables a las ranuras del NE. Si un número de ranura no figura en la lista, se supone que no hay modificación alguna de la asignación a dicha ranura. Si el valor de plugIn es la cadena de caracteres vacía no hay restricciones para el tipo de tarjeta que puede insertarse en la ranura y si posteriormente se quita una unidad enchufable de la ranura no se disparan las alarmas de extracción correspondientes. La entrada **alarmSeverityProfiles** identifica los perfiles destinados a configurar la gravedad de las alarmas individuales que ha de comunicar el NE. La entrada **thresholdDataProfiles** identifica los perfiles a utilizar en la configuración de valores umbral para generar las TCA del NE. La entrada **port** identifica el puerto PON del OLT si hay una modificación en la relación entre el ONT u ONU planificados y el puerto OLT. El puerto de entrada identifica el puerto ONU si hay un cambio en la relación entre el NT planificado y el puerto ONU. La entrada **newModelCode** proporciona una cadena de caracteres única que identifica el tipo de recurso de red. Este parámetro de entrada tiene interés sobre todo cuando el nodo modificado es un ONT o un NT. La entrada **newNEUserLabel** proporciona la nueva etiqueta de usuario que hay que aplicar al NE. La entrada **externalTime** proporciona la nueva hora de referencia actual para el NE. La entrada **administrationDomain** identifica el nuevo dominio al que se asigna el NE.

El valor devuelto es del tipo **void**.

9.2.1.4 deleteNode (*suprimir nodo*)

Esta operación suprime el NE del sistema de gestión del proveedor y cancela una petición anterior de suministro previo. Como consecuencia de esta operación se suprimen además todas las entidades gestionadas creadas automáticamente como resultado de la correspondiente operación buildNode. Se planteará la excepción **RemainingContainedManagedEntities** cuando haya otras entidades gestionadas contenidas que sigan presentes en las conexiones del servicio. Se suprimirán asimismo las reservas de anchura de banda asociadas a este nodo.

A continuación se muestra la signatura de la operación **deleteNode**:

```
void deleteNode (
            in ManagedEntityIdType managedEntityId)
            raises (UnknownNE, RemainingContainedManagedEntities, AccessDenied,
            RemainingReservations, RemainingSubnetworkConnections);
```

La entrada **managedEntityId** identifica el NE a suprimir en esta operación.

El valor devuelto es del tipo **void**.

9.2.1.5 modifyPort (*modificar puerto*)

Esta operación modifica los parámetros del puerto denominado, identificado por **physicalPathTPId**.

A continuación se muestra la signatura de la operación **modifyPort**:

```
void      modifyPort    (
               in ManagedEntityIdType physicalPathTPId,
               in NameSeqType newAlarmSeverityProfiles,
               in NameSeqType newThresholdDataProfiles,
               in NameSeqType newPortProfiles,
               in string newFrameFormat,
               in AdministrativeStateType administrativeState,
               in OpticalWaveLengthArraySeqType newOpticalWavelengthArray,
               in LoopbackLocationIdSeqType newLoopbackLocationIds,
               in unsigned long newInterfaceSpeed,
               in unsigned long aRCTimer)
               raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,
               InterfaceSpeedNotChangeable, ProfileSuspended );
```

La entrada **physicalPathTPId** identifica el puerto a modificar. La entrada **alarmSeverityProfiles** identifica los perfiles de configuración de la gravedad de las alarmas individuales que debe comunicar el NE. La entrada **thresholdDataProfiles** identifica los perfiles a utilizar cuando se configuran valores umbral para generar TCA del NE. El parámetro de entrada **newPortProfiles** identifica los perfiles utilizados para terminar el suministro del puerto. Son ejemplos de estos perfiles, entre otros, los siguientes: ATMNetworkAccessProfile, UNIInfo, EthernetProfile, CESServiceProfile, MACBridgeServiceProfile, LESServiceProfile, AAL1Profile, y AAL5Profile. La entrada **newFrameFormat** identifica un nuevo formato de trama que se terminará y generará en el puerto físico siempre que el formato de trama sea configurable. La entrada **administrativeState** especifica el valor modificado de este parámetro. La entrada **newLoopbackLocationIds** define nuevos identificadores de posición para el bucle asociado al puerto físico. La entrada **newInterfaceSpeed** identifica una nueva velocidad de interfaz del puerto físico siempre que éste sea configurable. El parámetro de entrada **aRCTimer** proporciona el tiempo no negativo (en segundos) que tiene el recurso de red para detectar que una señal es válida antes de disparar alarmas de comunicación en el puerto.

El valor devuelto es del tipo **void**.

9.2.1.6 buildPlugInUnit (*construir unidad enchufable*)

Esta operación construye una unidad enchufable en el sistema de gestión del proveedor como parte de las actividades de suministro previo. Como resultado de esta operación se crean automáticamente un conjunto de entidades gestionadas en el modelo de información de gestión mantenido por el sistema de gestión del proveedor. Dependiendo del tipo de unidad enchufable, se crearán automáticamente varios puntos de terminación. La asignación de ranuras del nodo contenedor se actualiza automáticamente para incluir la modificación solicitada por esta operación.

A continuación se muestra la signatura de la operación **buildPlugInUnit**:

```
ManagedEntityIdType buildPlugInUnit (
                        in ManagedEntityIdType nEId,
                        in NameType alarmSeverityProfile,
                        in UserLabelType plugInUnitUserLabel,
                        in string modelCode,
                        in AdministrativeStateType administrativeState,
                        in ManagedEntityIdType equipmentHolder)
                        raises (UnknownNE, DuplicateUserLabel,
                        AccessDenied, UnknownManagedEntity,
                        InvalidEquipmentCode, SlotAlreadyAssigned, UnknownSlot,
                        InvalidSlotAssignmentList, UnknownProfiles,
                        ProfileSuspended );
```

La entrada **nEId** proporciona un nombre único al NE que contiene la unidad enchufable a construir. La entrada **plugInUnitUserLabel** proporciona un identificador para la unidad enchufable a construir. La entrada **alarmSeverityProfile** identifica el perfil de configuración de las asignaciones de gravedad de las alarmas de avería del equipo relativas a la unidad enchufable. La entrada **modelCode** identifica el tipo de unidad enchufable. La entrada **administrativeState** especifica el valor inicial de este parámetro. La entrada **equipmentHolder** identifica la posición de la ranura en la que se insertará la unidad enchufable.

El valor devuelto del tipo **ManagedEntityIdType** proporciona un identificador único para el paquete de circuitos construido.

9.2.1.7 modifyPlugInUnit (*modificar unidad enchufable*)

Esta operación modifica los atributos de la unidad enchufable del sistema de gestión del proveedor.

A continuación se muestra la signatura de la operación **modifyPlugInUnit**:

```
ManagedEntityIdType     modifyPlugInUnit (
                    in ManagedEntityIdType plugInUnitId,
                    in NameType alarmSeverityProfile newAlarmSeverityProfile,
                    in string newModelCode,
                    in ManagedEntityIdType newEquipmentHolder,
                    in UserLabelType newPlugInUnitUserLabel,
                    in AdministrativeStateType newAdministrativeState)
                    raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,
                    InvalidEquipmentCode, SlotAlreadyAssigned, UnknownSlot,
                    InvalidSlotAssignmentList, InvalidUserLabelSyntax,
                    ProfileSuspended);
```

La entrada **plugInUnitId** identifica la entidad gestionada que ha de modificarse en esta operación. La entrada **newAlarmSeverityProfile** modifica el perfil de configuración de las asignaciones de gravedad de las alarmas de avería del equipo relativas a la unidad enchufable. La entrada **newModelCode** modifica el tipo de unidad enchufable. La entrada **newEquipmentHolder** modifica la ranura asignada a la unidad enchufable. La entrada **newPlugInUserLabel** proporciona una nueva etiqueta de usuario a la unidad enchufable. La entrada **newAdministrativeState** especifica el valor modificado de este parámetro.

El valor devuelto del tipo **ManagedEntityIdType** proporciona un identificador único para el paquete de circuitos modificado.

9.2.1.8 deletePlugInUnit (*suprimir unidad enchufable*)

Esta operación suprime una unidad enchufable del sistema de gestión del proveedor. Como consecuencia de esta operación, se suprimen además todas las entidades gestionadas creadas automáticamente como resultado de la correspondiente operación buildPlugInUnit. Se planteará la excepción **RemainingSubnetworkConnections** (**quedan conexiones de subred**) cuando sigan existiendo conexiones. Se utiliza para suprimir la información de suministro previo que ya no desea el operador.

A continuación se muestra la signatura de la operación **deletePlugInUnit**:

```
void deletePlugInUnit (
                    in ManagedEntityIdType plugInUnitId)
                    raises (UnknownManagedEntity, RemainingSubnetworkConnections,
                    AccessDenied, RemainingReservations);
```

La entrada **plugInUnitId** identifica la unidad enchufable a suprimir.

El valor devuelto es del tipo **void**.

9.2.1.9 buildProtectionGrouping (*construir grupo de protección*)

Esta operación construye un grupo de protección en el sistema de gestión del proveedor. Ningún puerto puede pertenecer a más de un grupo de protección y debe suministrarse antes de esta operación. Fuera del ámbito de IF1, hay un acuerdo entre proveedor y operador en cuanto a las medidas de protección válidas para el equipo del proveedor. Si hay un puerto protegido en la protectionUnitList, debe haber también un puerto protector como mínimo. Las características del trayecto físico de todos los puertos deben ser las mismas.

A continuación se muestra la signatura de la operación **buildProtectionGrouping**:

```
ManagedEntityIdType  buildProtectionGrouping (
                        in ProtectionParameterType protectionParameters,
                        in ProtectionUnitSeqType protectionUnitList)
                        raises (InvalidProtectionScheme, AccessDenied );
```

La entrada **protectionParameters** proporciona las características del esquema de protección. La entrada **protectionUnitList** identifica los puertos protector y protegido del recurso de red.

El valor devuelto es del tipo **ManagedEntityIdType** y proporciona un identificador único para la relación construida entre los puertos protegido y protector.

9.2.1.10 modifyProtectionParameters (*modificar parámetros de protección*)

Esta operación modifica el esquema de protección del grupo de protección identificado. Si se disminuyen m (= número de puertos protectores) o n (= número de puertos protegidos), será necesario que el OMS considere la invocación de modifyProtectionUnitList para evitar una excepción de esquema de protección no válido. Si se aumentan m o n, no es necesario invocar modifyProtectionUnitList.

A continuación se muestra la signatura de la operación **modifyProtectionParameters**:

```
void      modifyProtectionParameters (
                    in ManagedEntityIdType protectionGroupingId,
                    in ProtectionParameterType newProtectionParameters)
                    raises (UnknownManagedEntity, InvalidProtectionScheme,
                    AccessDenied);
```

La entrada **protectionGroupingId** identifica la entidad gestionada que ha de modificarse en esta operación. La entrada **newProtectionParameters** proporciona los parámetros de protección que sustituyen a los existentes.

El valor devuelto es del tipo **void**.

9.2.1.11 modifyProtectionUnitList (*modificar lista de unidades de protección*)

Esta operación permite efectuar adiciones y supresiones en la lista de puertos protectores y protegidos del grupo de protección identificado. Sólo se pueden suprimir todos los puertos protectores cuando se hayan suprimido todos los puertos protegidos. La adición de puertos protectores y protegidos no puede sobrepasar los límites de m y n especificados en la estructura de datos de protectionParameters asociada al grupo de protección. Ningún puerto puede pertenecer a más de un grupo de protección. Los puertos deben suministrarse antes de esta operación. Las características del trayecto físico de todos los puertos deben ser idénticas.

A continuación se muestra la signatura de la operación **modifyProtectionUnitList**:

```
void      modifyProtectionUnitList (
                    in ManagedEntityIdType protectionGroupingId,
                    in ProtectionUnitSeqType deltaProtectionUnitList,
                    in boolean addDeleteInd)
                    raises (UnknownManagedEntity, InvalidProtectionScheme,
                    AccessDenied);
```

La entrada **protectionGroupingId** identifica la entidad gestionada que ha de modificarse en esta operación. La entrada **deltaProtectionUnitList** proporciona las modificaciones de los puertos protegidos y protectores. El parámetro de entrada **addDeleteInd** muestra si la modificación es adición o supresión.

El valor devuelto es del tipo **void**.

9.2.1.12 deleteProtectionGrouping (*suprimir grupo de protección*)

Esta operación suprime un grupo de protección de puertos del sistema de gestión del proveedor. Como consecuencia de esta operación se suprimen además todas las entidades gestionadas creadas automáticamente como resultado de la correspondiente operación buildProtectionGrouping.

A continuación se muestra la signatura de la operación **deleteProtectionGrouping**:

```
void deleteProtectionGrouping (
                    in ManagedEntityIdType protectionGroupingId)
                    raises (UnknownManagedEntity, AccessDenied);
```

La entrada **protectionGroupingId** identifica el grupo de protección a suprimir.

El valor devuelto es del tipo **void**.

9.2.1.13 buildBridge (*construir puente*)

Esta operación construye un puente MAC en el sistema de gestión del proveedor. Todos los puertos deben suministrarse antes de esta operación. Todos los puertos UNI deben ser puertos LAN. Esta operación o no es necesaria si todos los puertos UNI LAN de un elemento de red pertenecen automáticamente al mismo puente y los valores por defecto de los parámetros del puente los establece el operador.

A continuación se muestra la signatura de la operación **buildBridge**:

```
ManagedEntityIdType  buildBridge (
                        in NameType mACBridgeProfile,
                        in ManagedEntityIdType uplinkPort,
                        in ManagedEntityIdSeqType uNIPortList)
```

```
                raises (UnknownProfiles, AccessDenied,
                UnknownManagedEntity, ProfileSuspended);
```

La entrada **mACBridgeProfile** proporciona las características del puente conformes con ANSI/IEEE Std 802.1D. La entrada **uplinkPort** identifica el puerto físico del OLT que hace de interfaz con la red troncal de la capa IP. Cuando no existe ninguna interfaz en el OLT, el valor de este parámetro es la cadena de caracteres vacía. La entrada **uNIPortList** identifica los puertos físicos UNI asociados al puente.

El valor devuelto es del tipo **ManagedEntityIdType** y proporciona un identificador único para el puente construido.

9.2.1.14 modifyBridgeProfile (*modificar perfil del puente*)

Esta operación modifica las características de la función de puente mediante el cambio de perfil de servicio del puente MAC asociado al puente.

A continuación se muestra la signatura de la operación **modifyBridgeProfile**:

```
void modifyBridgeProfile  (
                in ManagedEntityIdType bridgeId,
                in NameType newMACBridgeProfile)
                raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,
                ProfileSuspended);
```

La entrada **bridgeId** identifica la entidad gestionada que ha de modificarse en esta operación. La entrada **newMACBridgeProfile** proporciona el perfil de puente que sustituye al existente.

El valor devuelto es del tipo **void**.

9.2.1.15 modifyBridgePortList (*modificar lista de puertos del puente*)

Esta operación permite efectuar adiciones y supresiones en la lista de puertos UNI pertenecientes al grupo de puente. Puede darse el caso de que haya que suprimir todos los puertos UNI. Un puerto UNI no puede suprimirse si existe una conexión de servicio activa asociada al mismo. Si todos los puertos UNI LAN pertenecen forzosamente al mismo puente, esta operación ya no es necesaria.

A continuación se muestra la signatura de la operación **modifyBridgePortList**:

```
void modifyBridgePortList  (
                in ManagedEntityIdType bridgeId,
                in ManagedEntityIdSeqType deltaUNIPortList,
                in boolean addDeleteInd)
                raises (UnknownManagedEntity, RemainingSubnetworkConnections,
                AccessDenied);
```

La entrada **bridgeId** identifica la entidad gestionada que ha de modificarse en esta operación. La entrada **deltaUNIPortList** proporciona las modificaciones a los puertos UNI deseados. El parámetro de entrada **addDeleteInd** muestra si la modificación es adición o supresión.

El valor devuelto es del tipo **void**.

9.2.1.16 deleteBridge (*suprimir puente*)

Esta operación suprime el suministro de un puente del sistema de gestión del proveedor. Como consecuencia de esta operación se suprimen además todas las entidades gestionadas creadas automáticamente como resultado de la correspondiente operación buildBridge. Un puente no puede suprimirse si quedan conexiones de subred activas asociadas al mismo.

A continuación se muestra la signatura de la operación **deleteBridge**:

```
void deleteBridge (
            in ManagedEntityIdType bridgeId)
            raises (UnknownManagedEntity, AccessDenied,
            RemainingSubnetworkConnections);
```

La entrada **bridgeId** identifica el puente a suprimir.

El valor devuelto es del tipo **void**.

9.2.1.17 buildVPNetworkCTP (*construir un CTP de red VP*)

Esta operación construye un CTP de red del VP en el sistema de gestión del proveedor. El puerto que contiene este CTP de red del VP debe suministrarse antes de esta construcción. Esta operación se utiliza para construir CTP de red del VP destinados soportar las pruebas de la instalación. Se necesita tanto la operación de construir como la de suprimir.[3]

A continuación se muestra la signatura de la operación **buildVPNetworkCTP**:

```
ManagedEntityIdType  buildVPNetworkCTP (
                        in ManagedEntityIdType port,
                        in short vPI,
                        in NameType trafficDescriptorProfileName,
                        in ATMOverbookingFactorType overbookingFactor,
                        in UserLabelType   userLabel,
                        in SegmentEndpointIndType segmentEndpointInd)
                        raises (UnknownProfiles, AccessDenied,
                        UnknownManagedEntity, ParameterViolation,
                        ProfileSuspended);
```

La entrada **port** identifica la interfaz ATM del recurso de red. La entrada **vPI** proporciona el valor del índice. La entrada **trafficDescriptorProfileName** identifica el perfil descriptor de tráfico ATM asociado al CTP. El parámetro **overbookingFactor** proporciona el factor porcentual de desbordamiento que puede utilizarse en los algoritmos de control de admisión de llamadas de cualquier circuito virtual permanente que tenga el mismo valor VPI para el puerto interfaz ATM. El parámetro de entrada **userLabel** proporciona un nombre facilitado por el operador para el VPNetworkCTP. El parámetro de entrada **segmentEndpoint** identifica si el CTP es un punto extremo de segmento o no.

El valor devuelto es del tipo **ManagedEntityIdType** y proporciona un identificador único para el CTP construido.

9.2.1.18 deleteVPNetworkCTP (*suprimir CTP de red del VP*)

Esta operación suprime el suministro de un CTP de red del VP del sistema de gestión del proveedor. Como consecuencia de esta operación se suprimen además todas las entidades gestionadas creadas automáticamente como resultado de la correspondiente operación buildVPNetworkCTP.

A continuación se muestra la signatura de la operación **deleteVPNetworkCTP**:

```
void deleteVPNetworkCTP (
                    in ManagedEntityIdType vPNetworkCTP)
                    raises (UnknownManagedEntity, AccessDenied);
```

La entrada **vPNetworkCTP** identifica el puente a suprimir.

El valor devuelto es del tipo **void**.

---

[3] La operación de suprimir se describe en 9.2.1.18.

9.2.1.19 createdNodesGet (*obtener nodos creados*)

Esta operación recupera la lista de elementos de red construidos mediante la invocación de esta interfaz.

A continuación se muestra la signatura de la operación **createdNodesGet**:

```
ManagedEntityIdSeqType createdNodesGet () raises (AccessDenied);
```

No hay parámetros de entrada.

El valor devuelto es del tipo **ManagedEntityIdSeqType** y proporciona la lista de nodos que se han construido solicitando la interfaz Q834: Builder.

9.2.1.20 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema no tiene autorización de acceso al objeto de interfaz.

La excepción **DuplicateSerialNumber** (***número de serie duplicado***) se plantea cuando existe otro equipo del mismo tipo con este número de serie.

La excepción **DuplicateUserLabel** (***etiqueta de usuario duplicada***) se plantea cuando la etiqueta de usuario proporcionada en la petición se ha utilizado en otro NE o unidad enchufable. Dicho de otro modo, el sistema de gestión del proveedor supervisa la asignación de etiquetas de usuario a los NE y unidades enchufables en su jurisdicción de gestión.

La excepción **InterfaceSpeedNotChangeable** (***no es posible modificar la velocidad de la interfaz***) se plantea cuando el puerto físico no puede soportar la nueva velocidad de interfaz o si ésta no es configurable.

La excepción **InvalidEquipmentCode** (***código de equipo no válido***) se plantea cuando el código del equipo no se ajusta a la sintaxis acordada entre operador y proveedor.

La excepción **InvalidExternalTime** (***hora externa no válida***) se plantea cuando la hora externa especificada no es válida.

La excepción **InvalidProtectionScheme** (***esquema de protección no válido***) se plantea cuando el recurso de red no soporta los parámetros de protección especificados en el contexto del listado de puertos o si las unidades de protección son puertos con distintas características de trayecto físico.

La excepción **InvalidSerialNumSyntax** (***sintaxis del número de serie no válida***) se plantea cuando la sintaxis del número de serie proporcionado viola la sintaxis del proveedor.

La excepción **InvalidSlotAssignmentList** (***lista de asignación de ranuras no válida***) se plantea cuando el equipmentHolder designado no puede aceptar el tipo de unidad enchufable solicitado.

La excepción **InvalidUserLabelSyntax** (***sintaxis de etiqueta de usuario no válida***) se plantea cuando la etiqueta de usuario proporcionada viola las reglas de sintaxis de la empresa definidas por el operador e implementadas en el sistema de gestión del proveedor.

La excepción **ParameterViolation** (***violación de parámetros***) se plantea cuando el VPI está fuera de rango o duplicado.

La excepción **ProfileSuspended** (***perfil suspendido***) se plantea cuando el OMS o el operador hayan suspendido el uso en el sistema de gestión del proveedor de los perfiles nombrados en la invocación.

La excepción **RemainingContainedManagedEntities** (***quedan entidades gestionadas contenidas***) se plantea cuando quedan entidades gestionadas contenidas en la entidad gestionada a suprimir.

La excepción **RemainingReservations** (***quedan reservas***) se plantea cuando quedan reservas de recursos asociadas a la entidad gestionada a suprimir.

La excepción **RemainingSubnetworkConnections** (***quedan conexiones de subred***) se plantea cuando la entidad gestionada que se suprime contiene una o más conexiones de subred válidas.

La excepción **SlotAlreadyAssigned** (***la ranura ya está asignada***) se plantea cuando la ranura solicitada ya está suministrada.

La excepción **UnknownManagedEntity** (***entidad gestionada desconocida***) se plantea cuando la entidad gestionada identificada es desconocida para el sistema de gestión del proveedor.

La excepción **UnknownNE** (***NE desconocido***) se plantea cuando el NE identificado es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownProfiles** (***perfiles desconocidos***) se plantea cuando el nombre de perfil proporcionado es desconocido para el sistema de gestión del proveedor y no puede recuperarse del repositorio de objetos de perfil.

La excepción **UnknownSlot** (***ranura desconocida***) se plantea cuando la ranura solicitada es desconocida para el NE.

La excepción **UnknownSystemTimingSource** (***fuente de temporización desconocida***) si la fuente horaria externa es desconocida para el sistema de gestión del proveedor o para el NE.

La excepción **UnrecognizedVersion** (***versión no reconocida***) se plantea cuando la versión del equipo proporcionada no concuerda con los valores conocidos.

9.3 Módulo Q834Common

Este módulo contiene las definiciones de los tipos de datos utilizados por más de uno de los demás módulos IDL de esta Recomendación. Proporciona asimismo tres grupos de constantes. Muchos de los tipos de datos mencionados en Q834Common provienen de la Rec. UIT-T X.780. El aspecto más importante de este fichero IDL es la definición de ManagedEntityIdType. La definición se extrae del fichero IDL y se muestra a continuación para su estudio.

```
struct NamingComponentType {
     string type;  // managed entity type
     string id;
};

typedef sequence<NamingComponentType> RDNType;
typedef sequence<RDNType> RDNSeqType;
typedef sequence<NameType>  NameSeqType;

enum IdType {
     none,
     x780_fineGrained,
     x780_coarseGrained
};

typedef RDNType MEIdType;

struct ManagedEntityIdType {
     IdType id;
     MEIdType mEId;
};
```

ManagedEntityIdType es el valor devuelto en muchas de las operaciones que aparecen en esta Recomendación. Al examinar la definición de este tipo de datos (de arriba a abajo) se puede observar que el sistema de gestión del proveedor devuelve un valor que proporciona una referencia a un objeto gestionado de textura fina, a una fachada de textura gruesa o a un identificador RDN de una estructura de datos del sistema de gestión del proveedor. En los tres casos la sintaxis fundamental para el mEId es una secuencia de pares de componentes de denominación que es la

utilizada por la Rec. UIT-T X.780 para los objetos gestionados y la utilizada por la Rec. UIT-T X.780.1 para las fachadas.

9.3.1 Módulo ProbableCauseConst e interfaz ProbableCause

La definición de este módulo se ajusta al modelo especificado en la Rec. UIT-T X.780 y utiliza la misma sintaxis para los valores de causa probable. Se proporcionan valores específicos de la tecnología además de los valores definidos en la Rec. UIT-T X.780.

9.3.2 Interfaz MonitoringParameter (*parámetro de supervisión*)

Esta interfaz proporciona los nombres de los parámetros supervisados de calidad de funcionamiento y tráfico a utilizar en las interfaces AlarmEventSupplier, ImpairmentPersistence y ProfileConsumer. Los valores de MonitoringParameter se expresan como cadenas de caracteres y son partes de datos filtrables de un evento estructurado.

9.3.3 Interfaz RecordSetType (*tipo de conjunto de registros*)

Esta interfaz proporciona los valores de los datos a utilizar para el tipo de datos recordset por las interfaces ReportController y RecordSetMgr. Los valores del tipo recordset se expresan como valores de precisión corta sin signo. Los valores 1-99 se reservan para HistoryDataType.

9.3.4 Interfaz PhysicalLayerLoopback (*bucle de la capa física*)

Esta interfaz proporciona los valores de datos a utilizar para el tipo de datos loopbackTestType por la interfaz TestActionPerformer. Los valores del tipo recordset se expresan como valores de precisión corta sin signo.

9.4 Módulo ControlArchive (*archivo de control*)

El sistema de gestión del proveedor proporciona la funcionalidad de gestionar registros históricos de grupos específicos de eventos, entre ellos el borrado del contenido de los registros históricos. El usuario privilegiado puede crear, inicializar, suspender, reanudar y suprimir registros históricos de eventos. El sistema de gestión del proveedor proporciona asimismo la funcionalidad de controlar el archivo a corto plazo de informes de supervisión de la calidad de funcionamiento y de supervisión del tráfico, incluido el borrado del contenido de estos conjuntos de registros. Esta función incluye asimismo la generación de informes de estado de los registros históricos actuales o de los conjuntos de registros estadísticos. La mayor parte de los archivos de corto plazo del sistema de gestión del proveedor se crean automáticamente cuando el sistema de gestión del proveedor se ejemplifica por primera vez. Los nombres de estos archivos, identificados por las sintaxis ManagedEntityIdType, puede proporcionarlos al operador el proveedor o ser obtenidos por invocación de la operación recordSetListGet con el valor de creationModeType igual a "initialList" e invocaciones reiteradas de la operación getStatusAttributes para cada conjunto de registros identificado en la primera operación.

9.4.1 Interfaz RecordSetMgr (*gestor del conjunto de registros*)

9.4.1.1 createLog (*crear registro histórico*)

La operación CreateLog se utiliza para crear un conjunto de registros en el sistema de gestión del proveedor con el fin de archivar información sobre eventos.

La signatura de **createLog** es la siguiente:

```
ManagedEntityIdType createLog (
              in UserLabelType recordSetUserLabel,
              in AdministrativeStateType administrativeState,
              in NameType filterName,
              in FullActionType fullAction,
              in MaxSizeType maxSize,
```

```
            in SizeThresholdType sizeThreshold)
            raises (RecordSetExists, DuplicateUserLabel, AccessDenied);
```

El parámetro de entrada **recordSetUserLabel** identifica unívocamente el conjunto de registros (*recordSet*) en el sistema de gestión del proveedor. El parámetro de entrada **administrativeState** especifica si el nuevo registro histórico se inicializa para registrar información de eventos. El parámetro de entrada **filterName** indica los criterios de introducción que determinan qué notificaciones de eventos se han de grabar en el registro histórico que se crea. Si el **filterName** no es reconocido por el sistema de gestión del proveedor, busca en el IOR el filterObject utilizando **filterName**, consultando el directorio del servicio de denominación. Utilizando el IOR, el sistema puede recuperar los detalles de los criterios de entrada de la interfaz CosNotifyFilter existente en el servicio de notificación. El parámetro de entrada **fullAction** especifica el comportamiento del recordSet cuando éste alcanza su tamaño máximo. El parámetro de entrada **maxSize** especifica el tamaño máximo del recordSet. El parámetro de entrada **sizeThreshold** especifica el umbral del tamaño del RecordSet que provoca que el sistema de gestión del proveedor genere una alarma o evento.

El valor devuelto es del tipo **ManagedEntityIdType** y proporciona el identificador del registro histórico creado en esta operación.

9.4.1.2 createArchive (*crear archivo*)

La operación CreateArchive se utiliza para crear un recordSet para almacenar datos históricos. La operación de archivo se detendrá automáticamente cuando alcance su tamaño máximo.

La signatura de **createArchive** es la siguiente:

```
ManagedEntityIdType createArchive (
             in UserLabelType recordSetUserLabel,
             in AdministrativeStateType administrativeState,
             in RecordKindType recordKind,
             in MaxSizeType maxSize)
             raises (RecordSetExists, DuplicateUserLabel, AccessDenied);
```

El parámetro **recordSetUserLabel** identifica unívocamente el conjunto de registros (*recordSet*) en el sistema de gestión del proveedor. El parámetro de entrada **administrativeState** especifica si el nuevo archivo se inicializa para el almacenamiento de los registros adecuados. El parámetro de entrada **recordKind** identifica el tipo de registro a almacenar en el conjunto de registros. El parámetro de entrada **maxSize** especifica el tamaño máximo del recordSet.

El valor devuelto es del tipo **ManagedEntityIdType** y proporciona el identificador del archivo creado en esta operación.

9.4.1.3 getStatusAttributes (*obtener atributos de estado*)

El operador u OMS puede, en todo momento observar el estado actual de un archivo a corto plazo.

La signatura de **getStatusAttributes** es la siguiente:

```
RecordSetStatusType getStatusAttributes (
                  in ManagedEntityIdType recordSetId)
                  raises (AccessDenied, UnknownRecordSet);
```

El parámetro de entrada **recordSetId** identifica unívocamente el recordSet en el sistema de gestión del proveedor.

El valor devuelto es del tipo **RecordSetStatusType** y contiene el estado actual del recordset.

9.4.1.4 suspendArchive (*suspender archivo*)

Una vez creado e inicializado para su utilización el archivo de corto plazo, el OMS puede suspender su utilización.

La signatura de **suspendArchive** es la siguiente:

```
void suspendArchive (
                        in ManagedEntityIdType recordSetId)
                        raises (AccessDenied, UnknownRecordSet);
```

El parámetro de entrada **recordSetId** identifica unívocamente el recordSet en el sistema de gestión del proveedor.

El valor devuelto es del tipo **void**.

9.4.1.5 resumeArchive (*reanudar archivo*)

Esta operación reanuda el registro en un archivo o inicializa el registro en un conjunto de registros que se ha construido con estado bloqueado.

La signatura de **resumeArchive** es la siguiente:

```
void resumeArchive (
                        in ManagedEntityIdType recordSetId)
                        raises (UnknownRecordSet, AccessDenied);
```

El parámetro de entrada **recordSetId** identifica unívocamente el recordSet en el sistema de gestión del proveedor.

El valor devuelto es del tipo **void**.

9.4.1.6 deleteArchive (*suprimir archivo*)

Esta operación suprime un archivo del sistema de gestión del proveedor.

La signatura de **deleteArchive** es la siguiente:

```
void deleteArchive (
            in ManagedEntityIdType recordSetId )
            raises (UnknownRecordSet, AccessDenied );
```

El parámetro de entrada **recordSetId** identifica el nombre que el usuario privilegiado desea utilizar para identificar el registro histórico que ha de crearse en las interacciones posteriores.

El valor devuelto es del tipo **void**.

9.4.1.7 purgeArchive (*purgar archivo*)

Esta operación suprime la información contenida en un archivo específico. Este archivo, no obstante, puede continuar su grabación.

La signatura de **purgeArchive** es la siguiente:

```
void purgeArchive (
              in ManagedEntityIdType recordSetId )
              raises (UnknownRecordSet, AccessDenied );
```

El parámetro de entrada **recordSetId** identifica el nombre que el usuario privilegiado desea utilizar para identificar al registro histórico que ha de crearse en las interacciones posteriores.

El valor devuelto es del tipo **void**.

9.4.1.8 selectRecords (*seleccionar registros*)

Tras la creación de un archivo, el OMS puede seleccionar ciertos registros del mismo.

La signatura de **selectRecords** es la siguiente:

```
RecordSeqType selectRecords (
                  in FilterType selectionFilter,
```

```
                    in ManagedEntityIdType recordSetId)
                    raises (UnknownRecordSet, Timeout, NoSuchRecords,
                    AccessDenied, TooManyRecords);
```

El parámetro de entrada **recordSetId** identifica el nombre que el usuario privilegiado desea utilizar para identificar el registro histórico que ha de crearse en las interacciones posteriores. El parámetro de entrada **selectionFilter** se define por un conjunto específico de registros definidos en la estructura de datos.

El valor devuelto es del tipo **RecordSeqType** y proporciona la información solicitada.

9.4.1.9 recordSetListGet (*obtener lista de conjuntos de registros*)

Esta operación permite que un OMS obtenga un listado completo de los conjuntos de registros gestionados por el sistema de gestión del proveedor.

La signatura de **recordSetListGet** es la siguiente:

```
ManagedEntityIdSeqType recordSetListGet (
                        in CreationModeType     creationMode)
                        raises (AccessDenied);
```

El parámetro de entrada **creationMode** identifica el modo de creación del archivo, es decir, si lo creó un operador o se creó automáticamente como parte de la inicialización del sistema de gestión del proveedor.

El valor devuelto es del tipo **ManagedEntityIdSeqType** y lista los nombres de los archivos de corto plazo del sistema de gestión del proveedor.

9.4.1.10 changeUserLabel (*modificar etiqueta de usuario*)

Tras la creación de un archivo de corto plazo, el OMS puede modificar la etiqueta de usuario asignada al mismo.

La signatura de **changeUserLabel** es la siguiente:

```
void changeUserLabel(
            in ManagedEntityIdType recordSetId,
            in UserLabelType newUserLabel)
            raises (UnknownRecordSet, AccessDenied, DuplicateUserLabel);
```

El parámetro de entrada **recordSetId** identifica el archivo. El parámetro de entrada **newUserLabel** identifica el nuevo nombre para el archivo especificado.

El valor devuelto es del tipo **void**.

9.4.1.11 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema no tiene autorización de acceso al objeto de interfaz.

La excepción **DuplicateUserLabel** (***etiqueta de usuario duplicada***) se plantea cuando la etiqueta de usuario proporcionada en la petición ya se ha utilizado para etiquetar otro archivo. Dicho de otra forma, el sistema de gestión del proveedor supervisa la asignación de etiquetas de usuario a los conjuntos de registros en su jurisdicción de gestión.

La excepción **LockedAlready** (***ya está bloqueado***) se plantea cuando el valor establecido del administrativeState es "locked" (*bloqueado*).

La excepción **NoSuchRecords** (***no existen estos registros***) se plantea cuando en los conjuntos de registros designados no existe registro alguno que cumpla los criterios de selección.

La excepción **RecordSetExists** (***existe conjunto de registros***) se plantea cuando el archivo definido por los parámetros de la petición de creación ya existe en el sistema de gestión del proveedor.

La excepción **Timeout** (***fin de temporización***) se plantea cuando la duración del proceso alcanza un límite temporal definido por el sistema sin que el proceso pueda completarse.

La excepción **TooManyRecords** (***demasiados registros***) se plantea cuando el número de registros seleccionados en un proceso de recuperación provoca una respuesta a la petición que sobrepasa un tamaño predeterminado.[4]

La excepción **UnknownRecordSet** (***conjunto de registros desconocido***) se plantea cuando el conjunto de registros designado es desconocido para el sistema de gestión del proveedor.

9.5 Módulo SoftwareDownload (*descarga de soporte lógico*)

El proceso de descarga de soporte lógico consta de cuatro fases: entrega, distribución, instalación (compromiso) y activación. El sistema de gestión del proveedor soporta estas cuatro etapas para la descarga de programas informáticos genéricos, actualizaciones de soporte lógico y modificaciones de mantenimiento de soporte lógico (parches) con destino a los NE de este módulo. El sistema de gestión del proveedor puede aceptar solicitudes que afecten a uno o varios NE simultáneamente. Todas las actividades de gestión de soporte lógico pueden planificarse individualmente. Cualquier petición de descarga de soporte lógico genérico viene acompañada de la correspondiente acreditación de seguridad mediante la cual se permite al sistema de gestión del proveedor comunicarse con el servidor fuente. La utilización de una u otra acreditación de seguridad depende de la implementación: si la carga de soporte lógico es residente en el EMS y éste se la impone (mediante transferencia de ficheros) al NE, la acreditación es para el NE. Si la carga de soporte lógico es residente en el EMS y es el NE el que las demanda, la acreditación es para el EMS. Si la carga de soporte lógico es residente en un repositorio independiente, la acreditación de seguridad es para el repositorio independiente y el NE demanda la carga del repositorio.

Las peticiones de actividad de gestión de soporte lógico pueden provocar la creación de un objeto de seguimiento de gestión de soporte lógico. El ciclo de vida de este objeto debe ampliarse hasta completar todas las actividades asociadas con éxito o sin él. El objeto sólo se suprime tras un periodo de retención predefinido, acordándose su duración fuera del ámbito de esta interfaz. El periodo de retención del objeto de seguimiento debe ser lo suficientemente generoso como para incluir cualquier actividad de reversión deseada. Si el sistema de gestión del proveedor hubiera suprimido automáticamente el objeto de seguimiento, la actividad de reversión se contemplaría como actividad de descarga de una carga previa de soporte lógico. Además, el OMS puede suprimir este objeto de seguimiento antes de su terminación automática, cuando convenga.

El sistema de gestión del proveedor crea un registro histórico para las actividades de distribución de soporte lógico, entre ellas la entrega, compromiso y activación, con independencia del resultado de la actividad. El registro contiene los resultados detallados para cada objetivo implicado, incluidos los instantes de arranque/parada y la indicación de éxito o fracaso.

El sistema de gestión del proveedor mantiene un listado actualizado de todas las actividades informáticas en curso.

Este módulo incluye asimismo soporte de recuperación de la versión del soporte físico y soporte lógico de los NE instalados. Asimismo es posible verificar la posibilidad de que un conjunto de soporte lógico pueda descargarse a un NE o a una unidad enchufable específica.

9.5.1 Interfaz DownloadMgr (*gestor de descargas*)

9.5.1.1 deliverDistSWGlobal (*entregar distribución de soporte lógico mundial*)

Esta operación solicita al sistema de gestión del proveedor que descargue soporte lógico genérico de la máquina fuente de soporte lógico para actualizar el soporte lógico o aplicar cambios de

[4] Este tamaño deben acordarlo, previamente, el proveedor y el operador.

mantenimiento (parches) de soporte lógico a los NE. El sistema de gestión del proveedor puede aceptar solicitudes para un NE o para varios simultáneamente. Esta operación se realiza sin garantías, es decir que el sistema de gestión del proveedor intenta llevar a cabo la entrega y distribución a todos los objetivos especificados. En su caso, continuará intentando entregar y distribuir el soporte lógico a todos los objetivos nombrados aun en el caso de que encuentre un fallo en cualquier de los procesos en un objetivo específico. Se utiliza el registro histórico de compleción de la actividad para registrar los éxitos y fracasos de los intentos de entrega y distribución. El SoftwareDownloadTrackingObject (*objeto de seguimiento de descarga de soporte lógico*) continúa estando disponible para las actividades de compromiso, intentos de activación y aquellas que sólo se intentan con los objetivos en los que ha tenido éxito la entrega y distribución. El SoftwareDownloadTrackingObject se retiene más allá de la etapa de activación con éxito (entre otros motivos para permitir las actividades de reversión, en su caso).

Posteriormente, si se añade un nuevo ONT, se repara una conexión de fibra rota, o se instala una nueva unidad enchufable y la carga de soporte lógico activo es, en cada uno de estos casos, diferente de la especificada en esta operación, el sistema de gestión del proveedor (o recurso de red), se encargará de la actualización del soporte lógico automáticamente.

La signatura de **deliverDistSWGlobal** es la siguiente:

```
SoftwareDownloadTrackingObjectIdType deliverDistSWGlobal (
                        in FilenameSeqType softwareSet,
                        in DCNAddressType softwareSourceAddr,
                        in UserIdType userId,
                        in PasswordType    password,
                        in ManagedEntityIdSeqType deliverDistTargets)
                        raises (CommFailure, UnrecognisedTarget,
                        InsufficientMemory, SoftwareLoadHWMismatch,
                        SourceUnreachable, UnknownSoftwareLoad, Timeout,
                        AccessDenied, DeniedAccess);
```

El parámetro de entrada **softwareSet** identifica la carga de soporte lógico y su posición (nombres de los ficheros y rutas de acceso completos) desde donde debe descargarse. El parámetro de entrada **softwareSourceAddr** proporciona la dirección RCD del servidor en el que reside el conjunto de soporte lógico. Los parámetros de entrada **userId** y **password** proporcionan el mecanismo de entrada en comunicación a la softwareSource (suponiendo que se necesita esta acreditación de seguridad). El parámetro de entrada **deliverDistTargets** indica la lista de OLT a los que hay que entregar el soporte lógico.

El valor devuelto es del tipo **SoftwareDownloadTrackingObjectIdType** y proporciona una referencia a utilizar cuando se intenta comprometer, activar o verificar el estado del proceso de entrega y distribución. El sistema de gestión del proveedor anota en el registro histórico el resultado de la descarga del soporte lógico.

9.5.1.2 deliverDistSWSpecific (*peculiaridades de la entrega y distribución de soporte lógico*)

Esta operación coincide con deliverDistSWGlobal salvo en el alcance que aquí se circunscribe a un único NE, unidad enchufable o ranura descrito en el parámetro de entrada distributionTarget (*objetivo de distribución*).

La signatura de la operación correspondiente a **deliverDistSWSpecific** es la siguiente:

```
SoftwareDownloadTrackingObjectIdType deliverDistSWSpecific (
                        in FilenameSeqType softwareSet,
                        in DCNAddressType softwareSourceAddr,
                        in UserIdType userId,
                        in PasswordType password,
                        in TargetType deliverDistTarget)
                        raises (CommFailure, UnrecognisedTarget,
                        InsufficientMemory, SoftwareLoadHWMismatch,
```

```
            SourceUnreachable, UnknownSoftwareLoad, Timeout,
            AccessDenied, DeniedAccess);
```

El parámetro de entrada **softwareSet** identifica el soporte lógico a descargar y su posición (nombres de los ficheros y rutas de acceso completos). El parámetro de entrada **softwareSourceAddr** proporciona la dirección RCD del servidor en el que reside el conjunto de soporte lógico. Los parámetros de entrada **userId** y **password** proporcionan los mecanismos de entrada en comunicación con la softwareSource (suponiendo que se necesite esta acreditación de seguridad). El parámetro de entrada **deliverDistTarget** indica el objetivo específico a nivel de NE, unidad enchufable o ranura. En el cuadro 3 se describen los componentes del valor de este parámetro.

Cuadro 3/Q.834.4 – Detalles de deliverDistTarget

Nombre del campo	Descripción
containingSystem (*sistema contenedor*)	El sistema se identifica mediante el Id de entidad gestionada del OLT de cabecera.
containingNE (*NE contenedor*)	Identifica el recurso de red de destino. Cuando su valor es la secuencia vacía, la operación distribuye el soporte lógico a todos los NE del sistema.
plugInUnitType (*tipo de unidad enchufable*)	Identifica el tipo de unidad enchufable. Cuando su valor es la cadena de caracteres vacía, el objetivo de distribución son todos los tipos de unidad enchufable adecuados del NE contenedor.
slot (*ranura*)	Identifica la ranura. Cuando su valor es la secuencia de caracteres vacía, se considera adecuada cualquier ranura.

El valor devuelto es del tipo **SoftwareDownloadTrackingObjectIdTypeId** (***Id de tipo de Id de objeto de seguimiento de descarga de soporte lógico***) y proporciona una referencia de la clave de correlación que hay que utilizar al intentar comprometer, activar o verificar el estado del proceso de entrega y distribución. El sistema de gestión del proveedor anota el resultado de la descarga de soporte lógico en el registro histórico.

9.5.1.3 deleteSoftwareDownloadTrackingObject (*suprimir objeto de seguimiento de descarga de soporte lógico*)

Esta operación permite al OMS notificar al sistema de gestión del proveedor que el objeto especificado de seguimiento de soporte lógico ya no es necesario.

A continuación se facilita la signatura de la operación **deleteSoftwareDownloadTrackingObject**:

```
void deleteSoftwareDownloadTrackingObject  (
            in SoftwareDownloadTrackingObjectIdType   id)
            raises (UnknownSoftwareDownloadTrackingObject, AccessDenied,
            SoftwareTrackingObjectInUse);
```

El parámetro de entrada **id** identifica el objeto de seguimiento de soporte lógico.

El valor devuelto es del tipo **void**.

9.5.1.4 commit (*comprometer*)

Esta operación solicita al sistema de gestión del proveedor que instale (comprometa) el soporte lógico descargado en las posiciones objetivo.

A continuación se facilita la signatura de la operación **commit**:

```
void commit (
      in SoftwareDownloadTrackingObjectIdType id,
      in TargetType commitTarget)
      raises (InstallationFailure, UnknownSoftwareDownloadTrackingObject,
      AccessDenied, UnrecognisedTarget);
```

El parámetro de entrada **id** proporciona una referencia al objeto de seguimiento de descarga de soporte lógico. El parámetro de entrada **commitTarget** indica el objetivo específico a nivel de NE, unidad enchufable o ranura. El commitTarget puede ser un subconjunto del objetivo original.

El valor devuelto es del tipo **void**.

9.5.1.5 activate (activar)

Esta operación activa el soporte lógico instalado en las posiciones objetivo.

A continuación se facilita la signatura de la operación **activate**:

```
void activate (
            in SoftwareDownloadTrackingObjectIdType id,
            in TargetType activateTarget)
            raises (UnknownSoftwareDownloadTrackingObject,
            SoftwareNotYetInstalled, ActivationFailure, AccessDenied,
            UnrecognisedTarget);
```

El parámetro de entrada **id** proporciona una referencia al objeto de seguimiento de descarga de soporte lógico. El parámetro de entrada **activateTarget** indica el objetivo específico a nivel de NE, unidad enchufable o ranura. El activateTarget puede ser un subconjunto del objetivo original.

El valor devuelto es del tipo **void**.

9.5.1.6 revert (*revertir*)

Esta operación activa soporte lógico más antiguo en las posiciones objetivo. Sólo tiene sentido invocar la operación de revertir tras la activación de la carga de nuevo soporte lógico. El SoftwareDownloadTrackingObjectId (*Id del objeto de seguimiento de descarga de soporte lógico*) se refiere a la descarga de soporte lógico más reciente en el objetivo nombrado. Si la reversión se completa con éxito, la versión activa pasa a ser la de reserva.

A continuación se facilita la signatura de la operación **revert**:

```
void revert (
            in SoftwareDownloadTrackingObjectIdType id,
            in TargetType revertTarget)
            raises (UnknownSoftwareDownloadTrackingObject,
            SoftwareNotYetInstalled, ActivationFailure, AccessDenied,
            UnrecognisedTarget, InvalidSoftwareTrackingObject);
```

El parámetro de entrada **id** proporciona una referencia al objeto de seguimiento de descarga de soporte lógico. El parámetro de entrada **revertTarget** indica el objetivo específico a nivel de NE, unidad enchufable o ranura. El revertTarget puede ser un subconjunto del objetivo original.

El valor devuelto es del tipo **void**.

9.5.1.7 getStatus (*obtener estado*)

Esta operación solicita el estado de las actividades de soporte lógico.

A continuación se facilita la signatura de la operación **getStatus**:

```
DownloadStatusSeqType getStatus (
                        in SoftwareDownloadTrackingObjectIdType id)
                        raises (UnknownSoftwareDownloadTrackingObject,
                        AccessDenied);
```

El parámetro de entrada **id** proporciona una referencia al objeto de seguimiento de descarga de soporte lógico.

El valor devuelto es del tipo **DownloadStatusSeqType** (***tipo de secuencia de estado de descarga***) y proporciona el estado de desarrollo de todas las actividades de descarga supervisadas por el objeto de seguimiento de descarga de soporte lógico.

9.5.1.8 scheduleDeliverDist (*planificar distribución y entrega*)

Esta operación planifica la entrega y distribución de soporte lógico a los objetivos especificados. La distribución de soporte lógico en los objetivos especificados depende de la implementación del proveedor.

A continuación se facilita la signatura de la operación **scheduleDeliverDist**:

```
SoftwareDownloadTrackingObjectIdType scheduleDeliverDist (
                        in FilenameSeqType softwareSet,
                        in DCNAddressType softwareSourceAddr,
                        in UserIdType userId,
                        in PasswordType password,
                        in ManagedEntityIdSeqType deliverDistTargets,
                        in GeneralizedTimeType deliverDistStartTime)
                        raises (SoftwareLoadHWMismatch, AccessDenied,
                        InvalidStartTime );
```

El parámetro de entrada **softwareSet** identifica el soporte lógico a descargar y su posición (nombres de ficheros y rutas de acceso completos). El parámetro de entrada **softwareSourceAddr** proporciona la dirección RCD del servidor en el que reside el conjunto de soporte lógico. Los parámetros de entrada **userId** y **password** proporcionan el mecanismo de entrada en comunicación con la **softwareSource** (suponiendo necesaria esta acreditación). El parámetro de entrada **deliverDistTargets** indica la lista de OLT a los que hay que entregar el soporte lógico. El parámetro de entrada **deliverDistStartTime** proporciona la hora de comienzo planificada.

El valor devuelto es del tipo **SoftwareDownloadTrackingObjectIdType** (***tipo de Id de objeto de seguimiento de descarga de soporte lógico***) y proporciona una referencia al proceso planificado que puede utilizarse para hacer un seguimiento de la evolución del proceso o para cancelarlo.

9.5.1.9 scheduleCommit (*planificar compromiso*)

Esta operación planifica la instalación (compromiso) de soporte lógico en objetivos predeterminados.

A continuación se facilita la signatura de la operación **scheduleCommit**:

```
void scheduleCommit (
                    in SoftwareDownloadTrackingObjectIdType
                    deliverDistSoftwareDownloadTrackingObjectId,
                    in GeneralizedTimeType commitStartTime)
                    raises (UnknownSoftwareDownloadTrackingObject,
                    SoftwareNotYetInstalled, AccessDenied, InvalidStartTime );
```

El parámetro de entrada **deliverDistSoftwareDownloadTrackingObjectId** (*Id de objeto de seguimiento de descarga de soporte lógico de distribución y entrega*) proporciona una referencia al objeto de seguimiento de entrega de soporte lógico. El parámetro de entrada **commitStartTime** proporciona la hora de comienzo de la actividad planificada.

El valor devuelto es del tipo **void**.

9.5.1.10 scheduleActivate (*planificar activación*)

Esta operación planifica la activación del soporte lógico en objetivos predeterminados.

A continuación se facilita la signatura de la operación **scheduleActivate**:

```
void scheduleActivate (
                  in SoftwareDownloadTrackingObjectIdType id,
                  in GeneralizedTimeType activateStartTime)
                  raises (UnknownSoftwareDownloadTrackingObject,
                  SoftwareNotYetInstalled, AccessDenied, InvalidStartTime );
```

El parámetro de entrada **id** proporciona una referencia al objeto de seguimiento de entrega de soporte lógico. El parámetro de entrada **activateStartTime** proporciona la hora de comienzo de esta actividad.

El valor devuelto es del tipo **void**.

9.5.1.11 cancelScheduledSoftwareActivity (*cancelar actividad planificada de soporte lógico*)

Esta operación cancela todas las actividades posteriores de descarga de soporte lógico planificadas asociadas a este objeto de seguimiento.

A continuación se facilita la signatura de la operación **cancelScheduledSoftwareActivity**:

```
void cancelScheduledSoftwareActivity (
                        in SoftwareDownloadTrackingObjectIdType id)
                        raises (UnknownSoftwareDownloadTrackingObject,
                        ActivityCompleted, ActivityInProgress, AccessDenied);
```

El parámetro de entrada **id** proporciona una referencia al objeto de seguimiento de actividad de soporte lógico.

El valor devuelto es del tipo **void**.

9.5.1.12 scheduledSoftwareDownloadTrackingObjectListGet (*obtener lista de objetos de seguimiento de descarga de soporte lógico planificada*)

Esta operación recupera la lista de actividades planificadas pendientes de descarga de soporte lógico.

A continuación se facilita la signatura de la operación **scheduledSoftwareDownloadTrackingObjectListGet**:

```
SoftwareDownloadTrackingObjectIdSeqType
scheduledSoftwareDownloadTrackingObjectListGet () raises (AccessDenied);
```

No hay parámetros de entrada.

El valor devuelto es del tipo **SoftwareDownloadTrackingObjectIdSeqType** (***tipo de secuencia de Id de objeto de seguimiento de descarga de soporte lógico***) y proporciona la lista de actividades pendientes planificadas.

9.5.1.13 onDemandSoftwareDownloadTrackingObjectListGet (*obtener lista de objetos de seguimiento de descarga de soporte lógico a petición*)

Esta operación recupera la lista de actividades no planificadas pendientes de descarga de soporte lógico.

A continuación se facilita la signatura de la operación **onDemandSoftwareDownloadTrackingObjectListGet**:

```
SoftwareDownloadTrackingObjectIdSeqType
onDemandSoftwareDownloadTrackingObjectListGet () raises (AccessDenied);
```

No hay parámetros de entrada.

El valor devuelto es del tipo **SoftwareDownloadTrackingObjectIdSeqType** (***tipo de secuencia de Id de objeto de seguimiento de descarga de soporte lógico***) y proporciona la lista de actividades pendientes no planificadas.

9.5.1.14 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema no tiene autorización de acceso al objeto de interfaz.

La excepción **ActivityCompleted** (***actividad completada***) se plantea cuando la actividad de soporte lógico se ha ejecutado, no pudiendo por tanto cancelarse.

La excepción **ActivationFailure** (***fallo de activación***) se plantea cuando el proceso de activación de soporte lógico falló aun habiéndose instalado el soporte lógico (es decir habiéndose realizado con éxito el compromiso).

La excepción **ActivityInProgress** (***actividad en curso***) se plantea cuando la actividad de soporte lógico se ha iniciado y ya no puede cancelarse.

La excepción **CommFailure** (***fallo de comunicaciones***) se plantea cuando cae la RCD entre el sistema de gestión del proveedor y el OLT o se interrumpe la comunicación entre el OLT y el ONT fuente.

La excepción **DeniedAccess** (***denegación de acceso***) se plantea cuando el acceso al NE se deniega como resultado de restricciones del control de acceso.

La excepción **InstallationFailure** (***fallo de instalación***) se plantea cuando falla el proceso de instalación de soporte lógico.

La excepción **InsufficientMemory** (***memoria insuficiente***) se plantea cuando no hay memoria suficiente en el NE para cargar el soporte lógico.

La excepción **InvalidSoftwareTrackingObject** (***objeto de seguimiento de soporte lógico no válido***) se plantea cuando el objeto de seguimiento de soporte lógico referenciado no es el más reciente asociado a la instalación de una carga de soporte lógico en el NE.

La excepción **InvalidStartTime** (***hora de comienzo no válida***) se plantea cuando la hora de comienzo es anterior a la hora actual del sistema.

La excepción **SoftwareLoadHWMismatch** (***desadaptación entre el soporte físico y la carga de soporte lógico***) se plantea cuando el soporte lógico designado no puede cargarse en el soporte físico del equipo debido a que la versión del soporte físico no puede aceptar la carga de soporte lógico.

La excepción **SoftwareNotYetInstalled** (***soporte lógico pendiente de instalar***) se plantea cuando se solicita la activación y el soporte lógico no ha sido instalado todavía.

La excepción **SoftwareTrackingObjectInUse** (***objeto de seguimiento de soporte lógico utilizándose***) se plantea cuando hay actividades de soporte lógico pendientes supervisadas por este objeto y no puede suprimirse.

La excepción **SourceUnreachable** (***fuente no disponible***) se plantea cuando el servidor que guarda el soporte lógico a descargar no puede ser contactado por el OLT.

La excepción **Timeout** (***fin de temporización***) se plantea cuando la duración del proceso alcanza un límite temporal definido por el sistema antes de poder completar el proceso.

La excepción **UnknownSoftwareLoad** (***carga de soporte lógico desconocida***) se plantea cuando el conjunto de soporte lógico especificado no puede encontrarse.

La excepción **UnknownSoftwareDownloadTrackingObject** (***objeto de seguimiento de descarga de soporte lógico desconocido***) se plantea cuando el proceso de descarga de soporte lógico es desconocido para el sistema de gestión del proveedor.

La excepción **UnrecognizedTarget** (***objetivo no reconocido***) se plantea cuando el soporte lógico designado en el servidor de ficheros seguro es desconocido para el sistema de gestión del proveedor.

9.5.2 Interfaz VersionRepository (*repositorio de versiones*)

9.5.2.1 retrieveVersions (*recuperar versiones*)

Esta operación recupera toda la información de versiones (tanto de soporte lógico como de soporte físico) para un recurso de red.

A continuación se facilita la signatura de la operación **retrieveVersions**:

```
VersionsSeqType retrieveVersions (
                              in ManagedEntityIdType containingManagedEntityId)
                              raises (CommFailure, UnknownManagedEntity,
                              AccessDenied);
```

El parámetro de entrada **containingManagedEntityId** (***Id de la entidad gestionada contenedora***) identifica el recurso el red.

El valor devuelto es del tipo **VersionsSeqType** (***tipo de secuencia de versiones***) y proporciona un listado de las versiones de soporte físico y soporte lógico asociadas a este recurso de red.

9.5.2.2 validateNEVersion (*validar versión de NE*)

Esta operación solicita al sistema de gestión del proveedor que valide la compatibilidad del soporte lógico propuesto con el NE.

A continuación se facilita la signatura de la operación **validateNEVersion**:

```
boolean validateNEVersion (
            in ManagedEntityIdType managedEntityId,
            in VersionType proposedSoftware)
            raises (UnknownNE, AccessDenied);
```

El parámetro de entrada **managedEntityId** identifica el NE.

El parámetro de entrada **proposedSoftware** identifica el soporte lógico propuesto a validar.

El valor devuelto es del tipo **boolean**.

9.5.2.3 validatePlugInVersion (*validar versión de la unidad enchufable*)

Esta operación solicita al sistema de gestión del proveedor que valide la compatibilidad entre el soporte lógico propuesto y la unidad enchufable.

A continuación se facilita la signatura de la operación **validatePlugInVersion**:

```
boolean validatePlugInVersion (
                  in ManagedEntityIdType plugInUnitId,
                  in VersionType proposedSoftware)
                  raises (UnknownManagedEntity, AccessDenied);
```

El parámetro de entrada **plugInUnitId** (***Id de la unidad enchufable***) identifica la unidad enchufable.

El parámetro de entrada **proposedSoftware** (***soporte lógico propuesto***) identifica el soporte lógico propuesto a validar.

El valor devuelto es del tipo **boolean**.

9.5.2.4 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema no tiene autorización de acceso al objeto de interfaz.

La excepción **CommFailure** (***fallo de comunicaciones***) se plantea cuando cae la RCD entre el sistema de gestión del proveedor y el OLT o se interrumpe la comunicación entre el OLT y el ONT de origen.

La excepción **UnknownManagedEntity** (***entidad de gestión desconocida***) se plantea cuando el identificador de la unidad enchufable o puerto es desconocido para el sistema de gestión del proveedor.

9.6 Módulo EventPublisher (*publicador de eventos*)

Cuando el sistema de gestión del proveedor recibe información sobre la configuración procesada, la calidad de funcionamiento o eventos de averías, proporcionada por otros casos de empleo del sistema de gestión del proveedor, con sujeción a las reglas de publicación, el sistema de gestión del proveedor pone en cola y canaliza la información de eventos a todos los consumidores interesados, entre ellos los operadores y los OMS.

9.6.1 Interfaz AlarmEventSupplier (*proveedor de eventos de alarma*)

El objeto de esta interfaz es anunciar los eventos de alarma al sistema de gestión del operador mediante el servicio de notificación del OMG. Esta interfaz no tiene operaciones. No obstante proporciona la correspondencia de encabezamiento fijo así como las correspondencias de los datos filtrables para el objeto de evento estructurado que se utiliza para lanzar la información del evento por el canal de eventos del servicio de notificación del OMG. Ambos conjuntos de correspondencias deberán ajustarse a las directrices especificadas en la Rec. UIT-T X.780 para la interfaz de notificaciones.

En el encabezamiento fijo, se otorga a **domain_type** el valor "telecommunications", a **type_name** se le da el valor "Alarm", y a **event_name** se le da el valor de una cadena de caracteres constante que tiene uno de los valores siguientes: "Communications Alarm", "Environmental Alarm", "Equipment Alarm", "Processing Error Alarm" o "Quality of Service Alarm".

La correspondencia de los datos filtrables consta de un par de elementos. El primer componente del par es un identificador de cadena de caracteres correspondiente a un nombre de datos, mientras que el segundo es el valor de dicho elemento de datos. Los identificadores de cadena de caracteres son constantes que se definen en esta interfaz. Además, los pares de datos filtrables deben presentarse en un orden específico.

El orden de los elementos filtrables es el siguiente:

- AlarmEmittingMEId (*Id del NE emisor de la alarma*)
- EventTime (*hora del evento*)
- ProbableCause (*causa probable*)
- SpecificProblems (*problemas específicos*)
- PerceivedSeverity (*gravedad percibida*)
- ServiceAffectingInd (*Ind que afecta al servicio*)
- BackUpStatus (*estado de la copia de seguridad*)
- BackUpManagedEntityId (*Id de la entidad gestionada de respaldo*)

- ThresholdInfo (*información del umbral*)
- NotificationIdentifier (*identificador de la notificación*)
- CorrelatedNotifications (*notificaciones correlacionadas*)
- StateChangeDefinition (*definición de la modificación de estado*)
- AdditionalText (*texto adicional*)

El valor de ProbableCause tiene la sintaxis ProbableCauseType y toma uno de los valores específicos definidos en la Rec. UIT-T X.780 o q834_4::Q834Common::ProbableCauseConst.

El valor de ThresholdInfo tiene la sintaxis MonitoredParameterType y toma uno de los valores específicos definido en q834_4::Q834Common::MonitoredParameter. Se otorga la cadena de caracteres nula a este valor para todos los event_names excepto "QualityOfServiceAlarm".

La sintaxis e interpretación de los demás tipos de datos y la interpretación de su utilización aparecen en la Rec. UIT-T X.780.

9.6.2 SecurityEventSupplier (*proveedor de eventos de seguridad*)

El objeto de esta interfaz es anunciar los eventos de seguridad al sistema de gestión del operador mediante el servicio de notificación del OMG. Esta interfaz no tiene operaciones. No obstante proporciona la correspondencia de encabezamiento fijo así como las correspondencias de datos filtrables para el objeto de evento estructurado utilizado para introducir la información del evento por el canal de eventos del servicio de notificación del OMG. Ambos conjuntos de correspondencias deberán ajustarse a las directrices especificadas en la Rec. UIT-T X.780 para la interfaz de notificaciones.

En el encabezamiento fijo, se da a **domain_type** el valor "telecommunications", a **type_name** se le da el valor "SecurityEvent", y a **event_name** se le da el valor de una cadena de caracteres constante con uno de los siguientes valores: "IntegrityViolation", "OperationalViolation", "PhysicalViolation", "SecurityEventViolation" o "TimeDomainViolation".

La correspondencia en los datos filtrables consta de un par de elementos. El primer componente del par es un identificador de cadena de caracteres correspondiente a un nombre de datos mientras que el segundo es el valor de dicho elemento de datos. Los identificadores de cadena de caracteres son constantes que se definen en esta interfaz. Además los pares de datos filtrables deben presentarse en un orden específico.

El orden de los elementos filtrables es el siguiente:

- EventEmittingMEId (*Id del NE emisor del evento*)
- EventTime (*hora del evento*)
- SecurityAlarmCause (*causa de la alarma de seguridad*)
- SecurityAlarmDetector (*detector de la alarma de seguridad*)
- ServiceUser (*usuario de servicio*)
- ServiceProvider (*proveedor del servicio*)
- NotificationIdentifier (*identificador de la notificación*)
- CorrelatedNotifications (*notificaciones correlacionadas*)
- AdditionalText (*texto adicional*)

La sintaxis de los tipos de datos y la interpretación de su uso aparecen en la Rec. UIT-T X.780.

9.6.3 DiscoveryEventSupplier (*proveedor de eventos de descubrimiento*)

El propósito de esta interfaz es anunciar las modificaciones del equipo instalado al sistema de gestión del operador mediante el servicio de notificación del OMG. Esta interfaz no tiene

operaciones. No obstante, proporciona la correspondencia de encabezamiento fijo así como las correspondencias de datos filtrables para el objeto de evento estructurado utilizado para introducir la información del evento por el canal de eventos del servicio de notificación del OMG.

En el encabezamiento fijo, se da a **domain_type** el valor "telecommunications", a **type_name** se le da el valor "DiscoveryEvent", y a **event_name** se le da el valor de una cadena de caracteres constante con uno de los siguientes valores: "ManagedEntityCreation" o "ManagedEntityDeletion".

La correspondencia de los datos filtrables consta de un par de elementos. El primer componente de este par es un identificador de cadena de caracteres correspondiente a un nombre de datos mientras que el segundo es el valor de dicho elemento de datos. Los identificadores de cadena de caracteres son constantes que se definen en esta interfaz. Además, los pares de datos de filtrables deben presentarse en un orden específico.

El orden de los elementos filtrables para un event_name de "ManagedEntityCreation" es el siguiente:

- ManagedEntityType (*tipo de entidad gestionada*)
- EventTime (*hora del evento*)
- ManagedEntityAttributeValues (*valor de los atributos de la entidad gestionada*)
- NotificationIdentifier (*identificador de notificación*)
- CorrelatedNotifications (*notificaciones correlacionadas*)
- AdditionalText (*texto adicional*)

El valor de ManagedEntityType tiene la sintaxis de EquipmentType e indica el tipo de datos de inventario suprimidos. La interfaz define constantes para diversos tipos de NE así como plugInUnits y equipmentHolders.

El valor de EventTime tiene la sintaxis de GeneralizedTimeType y se refiere al momento en que la red detecta la condición de descubrimiento.

El valor de ManagedEntityAttributeValues tiene la sintaxis MEstruct. No obstante, el tipo de datos suministrado para este valor pertenece a un conjunto de estructuras de datos definidas en el módulo. El elemento de datos ManagedEntityType identifica el tipo de struct que se pasa en este valor en el objeto de evento estructurado.

El valor de NotificationIdentifier tiene la sintaxis NotificationIdentifierType y proporciona un número de secuencia de referencia al evento. El valor de CorrelatedNotifications tiene la sintaxis de CorrelatedNotificationType y proporciona una lista de números de referencia a otras notificaciones de eventos proporcionadas por el sistema de gestión del proveedor para condiciones de inventario asociadas. Si no hubiera notificaciones relacionadas se facilitaría el valor del conjunto vacío.

Por último, el valor de AdditionalText tiene la sintaxis de cadena de caracteres. Este elemento de datos proporciona una posición para pasar cualquier información textual de carácter diverso procedente del sistema de gestión del proveedor relativa a la condición de modificación de inventario. Si no hubiera información adicional, se pasaría la cadena de caracteres nula.

El orden de los elementos filtrables para un event_name de "ManagedEntityDeletion" es el siguiente:

- ManagedEntityType (*tipo de entidad gestionada*)
- EventTime (*hora del evento*)
- ManagedEntityAttributeValues (*valor de los atributos de la entidad gestionada*)
- NotificationIdentifier (*identificador de notificación*)
- CorrelatedNotifications (*notificaciones correlacionadas*)
- AdditionalText (*texto adicional*)

El valor de ManagedEntityType tiene la sintaxis de EquipmentType e indica el tipo de datos de inventario descubierto. La interfaz define constantes para diversos tipos de NE así como plugInUnits, equipmentHolders y soporte lógico.

El valor de EventTime tiene la sintaxis de GeneralizedTimeType y se refiere al momento en que la red detecta la condición de descubrimiento.

El valor de ManagedEntityAttributeValues tiene la sintaxis MEstruct. No obstante, el tipo de datos suministrado para este valor pertenece a un conjunto de estructuras de datos definidas en el módulo. El elemento de datos ManagedEntityType identifica el tipo de struct que se pasa en este valor en el objeto de evento estructurado.

El valor de NotificationIdentifier tiene la sintaxis NotificationIdentifierType y proporciona un número de secuencia de referencia al evento. El valor de CorrelatedNotifications tiene la sintaxis de CorrelatedNotificationType y proporciona una lista de números de referencia a otras notificaciones de eventos proporcionadas por el sistema de gestión del proveedor para condiciones de inventario asociadas. Si no hubiera notificaciones relacionadas se facilitaría el valor del conjunto vacío.

Por último, el valor de AdditionalText tiene la sintaxis de cadena de caracteres. Este elemento de datos proporciona una posición para pasar cualquier información textual de carácter diverso procedente del sistema de gestión del proveedor relativa a la condición de modificación de inventario. Si no hubiera información adicional, se pasaría la cadena de caracteres nula.

La inserción o supresión de una unidad enchufable crea siempre una notificación con independencia del estado de suministro en el que se encuentre la unidad enchufable.

9.7 Módulo MIBTransfer (*transferencia de la MIB*)

Este módulo gestiona el proceso de recogida de datos de configuración del sistema para restaurar un sistema en el evento de un fallo catastrófico. La recogida de datos de configuración puede realizarse en tiempo real o de forma programada. Proporciona asimismo información para los procesos de copia de seguridad y de recuperación en curso. El sistema de gestión del proveedor anota en el registro histórico el resultado de los procesos de copia de seguridad y restauración. Cualquier petición de hacer una copia de seguridad de los datos de configuración o de restaurarlos viene acompañada de una acreditación de seguridad mediante la cual se permite al sistema de gestión del proveedor o al OLT la comunicación con el servidor externo. En el transcurso de la copia de seguridad, todas las peticiones de prestación que afecten al sistema OLT designado se rechazan o bloquean. El sistema de gestión del proveedor suprime automáticamente los objetos de seguimiento de la transferencia una vez terminadas las transferencias de ficheros asociadas, anotándose los resultados (ya sean satisfactorios o insatisfactorios) en el registro histórico de compleción.

9.7.1 Interfaz MIBMover (*transferencia de la MIB*)

9.7.1.1 startBackup (*comenzar copia de seguridad*)

Esta operación inicia una copia de seguridad inmediata de los datos de configuración del sistema, desde un sistema y/o sistema de gestión del proveedor a un servidor de copia de seguridad de destino.

A continuación se muestra la signatura de la operación **startBackup**:

```
TransferTrackingObjectIdType startBackup (
                              in ManagedEntityIdType nEManagedEntityId,
                              in DCNAddressType destinationServerAddr,
                              in UserIdType userId,
                              in PasswordType password,
                              in FilenameType destinationFile,
                              in boolean overwriteExistingFile)
```

```
                                raises (UnknownNE, UnknownDestinationServer,
                                CommFailure, EquipmentFailure, DeniedAccess,
                                AccessDenied);
```

El parámetro de entrada **nEManagedEntityId** ***(Id de entidad gestionada del NE***) identifica el sistema objeto de la copia de seguridad. El parámetro de entrada **destinationServerAddr** identifica la dirección en la red de comunicación de datos del servidor de destino de la copia de seguridad. El parámetro de entrada **userId** identifica la clave de acceso del usuario para entrar en el servidor de destino. El parámetro de entrada **password** es la contraseña para acceder al servidor de destino. El parámetro de entrada **destinationFile** proporciona la ruta de acceso completa del fichero de copia de seguridad. Por último, el parámetro **overwriteExistingFile** indica si en el proceso de copia de seguridad se permite la sustitución de un fichero preexistente con la misma ruta de acceso.

El valor devuelto es del tipo **TransferTrackingObjectIdType** y proporciona una clave de correlación para observar el estado de desarrollo del proceso de copia de seguridad.

9.7.1.2 getBackupStatus (*obtener estado de la copia de seguridad*)

Esta operación permite recuperar el estado de un proceso de copia de seguridad.

A continuación se muestra la signatura de la operación **getBackupStatus**:

```
StatusAttributeSeqType getBackupStatus (
                        in TransferTrackingObjectIdType id)
                        raises (UnknownBackupProcess,AccessDenied);
```

El parámetro de entrada **id** identifica el proceso de copia de seguridad.

El valor devuelto es del tipo **StatusAttributeSeqType** (***tipo de secuencia del atributo de estado***) y proporciona el estado del proceso de copia de seguridad solicitado anteriormente.

9.7.1.3 scheduleBackup (*copia de seguridad planificada*)

Esta operación planifica procesos de copia de seguridad.

A continuación se muestra la signatura de la operación **scheduleBackup**:

```
TransferTrackingObjectIdType scheduleBackup(
                             in ManagedEntityIdType nEManagedEntityId,
                             in UserIdType userId,
                             in PasswordType password,
                             in UserLabelType schedulerName,
                             in DCNAddressType destinationServerAddr,
                             in FilenameType destinationFile,
                             in boolean overwriteExistingFile)
                             raises (UnknownNE, UnknownScheduler,
                             UnknownDestinationServer, InvalidScheduler,
                             AccessDenied);
```

El parámetro de entrada **nEManagedEntityId** identifica el OLT objeto de la copia de seguridad. El parámetro de entrada **schedulerName** es el nombre del planificador utilizado para la copia de seguridad. El parámetro de entrada **userId** identifica la clave de entrada del usuario en el servidor de destino. El parámetro de entrada **password** es la contraseña de acceso al servidor de destino. El parámetro **destinationServerAddr** indica la dirección en la RCD del servidor de destino sobre el que ha de efectuarse la copia de seguridad de los datos. El parámetro de entrada **destinationFile** proporciona la ruta de acceso completa del fichero de copia de seguridad. Finalmente, el parámetro **overwriteExistingFile** indica si se debe permitir a la copia de seguridad sustituir un fichero anterior que tenga la misma ruta de acceso.

El valor devuelto es del tipo **TransferTrackingObjectIdType** y proporciona una clave de correlación para ser utilizada cuando se intente efectuar un seguimiento del estado diferido, en algún momento posterior.

9.7.1.4 modifyBackupSchedule (*modificar planificación de la copia de seguridad*)

Esta operación modifica todos los procesos posteriores de copia de seguridad de un sistema con planificador. Esta operación no interrumpe el proceso de copia de seguridad en curso.

A continuación se muestra la signatura de la operación **modifyBackupSchedule**:

```
void modifyBackupSchedule (
              in TransferTrackingObjectIdType id,
              in UserLabelType newSchedulerName)
              raises (UnknownBackupProcess, AccessDenied,
              UnknownScheduler, InvalidScheduler);
```

El parámetro de entrada **id** identifica el proceso planificado a modificar. El parámetro de entrada **newSchedulerName** identifica los nuevos activadores temporales.

El valor devuelto es del tipo **void**.

9.7.1.5 cancelScheduledBackup (*cancelar copia de seguridad planificada*)

Esta operación cancela todos los procesos posteriores de copia de seguridad de un sistema con planificador. Esta operación no interrumpe el proceso de copia de seguridad en curso.

A continuación se muestra la signatura de la operación **cancelScheduledBackup**:

```
void cancelScheduledBackup (
            in TransferTrackingObjectIdType id)
            raises (UnknownBackupProcess, AccessDenied);
```

El parámetro de entrada **id** identifica el proceso planificado a cancelar.

El valor devuelto es del tipo **void**.

9.7.1.6 abortBackup (*abortar copia de seguridad*)

Esta operación aborta el proceso de copia de seguridad en curso, ya sea éste planificado o no. Esta operación no repercute en los procesos planificados posteriores.

A continuación se muestra la signatura de la operación **abortBackup**:

```
void abortBackup (
            in TransferTrackingObjectIdType id)
            raises (UnknownBackupProcess, CommFailure, EquipmentFailure,
            AccessDenied);
```

El parámetro de entrada **id** identifica el proceso de copia de seguridad a abortar.

El valor devuelto es del tipo **void**.

9.7.1.7 startRestore (*comenzar recuperación*)

La operación **startRestore** proporciona la funcionalidad de recuperación de un sistema con copia de seguridad de los datos de configuración.

A continuación se muestra la signatura de la operación **startRestore**. Los datos de configuración residen en un servidor seguro externo.

```
TransferTrackingObjectIdType startRestore (
                             in ManagedEntityIdType nEManagedEntityId,
                             in DCNAddressType sourceServerAddr,
                             in UserIdType userId,
                             in PasswordType password,
                             in FilenameType sourceFile)
                             raises (UnknownNE, CommFailure,
```

```
                                EquipmentFailure, UnknownSourceServer,
                                DeniedAccess, SoftwareLoadHardwareMismatch,
                                AccessDenied );
```

El parámetro de entrada **nEManagedEntityId** identifica el OLT a restaurar. El parámetro de entrada **sourceServerAddr** identifica la dirección en la red de comunicación de datos del servidor de destino de la copia de seguridad. El parámetro de entrada **userId** identifica la clave de acceso al servidor de destino. El parámetro de entrada **password** es la contraseña de acceso al servidor de destino. El parámetro **sourceFile** indica el fichero de copia de seguridad de los datos desde el que se efectuará la restauración.

El valor devuelto es del tipo **TransferTrackingObjectIdType** (***tipo de Id de objeto de seguimiento de transferencia***) y proporciona una clave de correlación a utilizar cuando se intenta hacer un seguimiento del estado de la restauración solicitada.

9.7.1.8 getRestoreStatus (*obtener estado de la restauración*)

Esta operación proporciona el estado de un proceso de restauración.

A continuación se muestra la signatura de la operación **getRestoreStatus**:

```
StatusAttributeSeqType getRestoreStatus (
                        in TransferTrackingObjectIdType id)
                        raises (UnknownRestoreProcess, AccessDenied);
```

El parámetro de entrada **id** identifica el proceso de restauración.

El valor devuelto es del tipo **StatusAttributeSeqType** y proporciona el estado de progreso del proceso de restauración.

9.7.1.9 transferTrackingObjectIdListGet (*obtener lista de ID de objetos de seguimiento de transferencia*)

Esta operación recupera de la MIB del sistema la lista de transferencias pendientes, tanto de copia de seguridad como de restauración.

A continuación se muestra la signatura de la operación **transferTrackingObjectIdListGet**:

```
TransferTrackingObjectIdSeqType transferTrackingObjectIdListGet ()
                    raises (AccessDenied);
```

No hay parámetros de entrada.

El valor devuelto es del tipo **TransferTrackingObjectIdSeqType** (***tipo de secuencia de Id de objeto de seguimiento de transferencia***) y proporciona la lista de transferencias pendientes de la MIB del sistema, tanto de copia de seguridad como de restauración.

9.7.1.10 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema no tiene autorización de acceso al objeto de interfaz.

La excepción **CommFailure** (***fallo de comunicaciones***) se plantea cuando cae la RCD entre el sistema de gestión del proveedor y el OLT o se interrumpe la comunicación entre el OLT y el ONT fuente.

La excepción **DeniedAccess** (***denegación de acceso***) se plantea cuando se deniega el acceso al NE.

La excepción **EquipmentFailure** (***fallo del equipo***) se plantea cuando el equipo desde el que se efectúa la copia de seguridad de los datos está averiado.

La excepción **InvalidScheduler** (***planificador no válido***) se plantea cuando el planificador en cuestión es inadecuado para esta operación o es obsoleto.

La excepción **SoftwareLoadHardwareMismatch** (***desadaptación entre el soporte físico y la carga de soporte lógico***) se plantea cuando el soporte lógico de origen no concuerda con el soporte físico de destino.

La excepción **UnknownBackupProcess** (***proceso de copia de seguridad desconocido***) se plantea cuando no se encuentra el proceso de copia de seguridad relacionado con el TransferTrackingObjectId en cuestión.

La excepción **UnknownDestinationServer** (***servidor de destino desconocido***) se plantea cuando no se encuentra la dirección IP del servidor de destino.

La excepción **UnknownNE** (***NE desconocido***) se plantea cuando el OLT es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownRestoreProcess** (***proceso de restauración desconocido***) se plantea cuando el proceso de restauración relacionado con el TransferTrackingObjectId es desconocido.

La excepción **UnknownScheduler** (***planificador desconocido***) se plantea cuando no se encuentra el nombre del planificador en cuestión.

La excepción **UnknownSourceServer** (***servidor de origen desconocido***) se plantea cuando el servidor de origen es desconocido para el sistema de gestión del proveedor.

9.8 Módulo PerformanceManager (*gestor de la calidad de funcionamiento*)

El sistema de gestión del proveedor se encargará de la activación y desactivación del proceso de comunicación de datos de la calidad de funcionamiento y de las mediciones de tráfico de los OLT instalados, desde los puntos de terminación individuales contenidos en los NE, a petición del operador o del OMS. Entre los requisitos del sistema de gestión del proveedor que figuran en esta Recomendación se encuentran asimismo el suministro de valores umbral y la descripción de la comunicación automática de mediciones de la calidad de funcionamiento al sistema de gestión del proveedor cuando se sobrepasan los umbrales. Cuando se presenta un conjunto de alertas de rebasamiento de umbral asociadas a una única condición de degradación de la calidad de funcionamiento, el sistema de gestión del proveedor deberá analizar y correlacionar los eventos de alerta de su dominio, en la medida de lo posible, determinar el origen del problema y anotar esta información en un registro histórico. Si en un determinado plazo se detectan varias incidencias motivadas por la misma causa de degradación, el sistema de gestión del proveedor preparará un registro de alarma de QoS para su publicación con destino a los consumidores interesados (operador u OMS). Si el sistema de gestión del proveedor no pudiera recoger todos los registros de datos históricos de todos los puntos de supervisión[5] de todos los elementos de red en su dominio de gestión, no sería necesario el control de informes. En tal caso se supondría que la información estaría siempre disponible.

9.8.1 Interfaz ImpairmentPersistence (*persistencia de la degradación*)

En esta Recomendación se supone y requiere que los objetos de perfil del tipo datos de umbral[6] sean grupos de valores umbral que puedan asociarse a un único tipo de punto de supervisión.

9.8.1.1 setSlidingWindowParameters (*establecer parámetros de la ventana deslizante*)

Esta operación establece los parámetros de ventana deslizante para uno, algunos o todos los parámetros supervisados en un NE. Estos valores se aplican a todos los puntos que supervisan los parámetros. Posteriormente, el sistema de gestión del proveedor genera una alarma QoS si un punto de supervisión detecta una TCA **persistenceMinimum** veces en **totConsecutiveIntvls** intervalos de supervisión consecutivos. Esta operación se efectúa sin garantías.

---

5 Los registros de datos históricos y los puntos de supervisión se definen en la Rec. UIT-T Q.834.1.

6 Del tipo 21 especificado en el módulo ProfileManager.

Se plantea la excepción CommFailure si el sistema de gestión del proveedor es incapaz de comunicarse con el NE identificado en la operación y si los valores de los parámetros de la ventana de deslizamiento se establecen directamente a partir de los recursos de la red.

Esta operación se utiliza para establecer valores por defecto para todo el sistema si los indicadores sysScopeInd tienen el valor TRUE y si el nEManagedEntityId es el identificador de un OLT. Esto significa que los valores de los parámetros de la ventana de deslizamiento han de aplicarse a todos los puntos de supervisión de cualquier componente del equipo que pertenezca actualmente, o que pueda pertenecer, al sistema que tenga la cabecera OLT nombrada. Si el OLT hubiera perdido las comunicaciones con cualquier recurso de red subyacente antes de la invocación de esta operación o durante ella, corresponde a la implementación del proveedor garantizar que los valores se aplican una vez restablecida la comunicación. Si el indicador sysScopeInd tuviera el valor TRUE y el nEManagedEntityId fuera el identificador de una ONU, los valores serían aplicables a la ONU y a todas las NT subyacentes. Conforme se añadan componentes de equipos al sistema que tenga la cabecera de la OLT u ONU, el sistema de gestión del proveedor establecerá automáticamente estos valores de parámetro.

Si sysScopeInd tuviera el valor FALSE, los valores de los parámetros de la ventana deslizante habrían de aplicarse únicamente al recurso indicado. Estos valores se seguirán aplicando a los componentes, existentes o posibles, del equipo del nodo.

A continuación se muestra la signatura de la operación **setSlidingWindowParameters**:

```
void setSlidingWindowParameters (
            in ManagedEntityIdType nEManagedEntityId,
            in MonitoredParameterSeqType monitoredParameterList,
            in short totConsecutiveIntvls,
            in short persistenceMinimum,
            in boolean sysScopeInd)
            raises (UnknownNE, UnknownParameters, IntervalCountTooLarge,
            AccessDenied, CommFailure);
```

El parámetro de entrada **nEManagedEntityId** identifica el elemento de red. El parámetro de entrada **monitoredParameterList** identifica los parámetros a supervisar. Los valores de los parámetros figuran en q834_4::Q834Common::MonitoredParameter. El parámetro de entrada **totConsecutiveIntvls** identifica el valor del total de intervalos consecutivos supervisados. El parámetro de entrada **persistenceMinimum** identifica el número mínimo de TCA para el monitoredParameter. La entrada **sysScopeInd** indica si los valores de la ventana deslizante deben aplicarse, o no, a todos los elementos de red subyacentes desde el inicial indicado en la operación.

El valor devuelto es del tipo **void**.

9.8.1.2 setSpecificSlidingWindowParameters (*establecer parámetros de ventana deslizante específicos*)

Esta operación es similar a la operación setSlidingWindowParameters excepto en que el ámbito de asignación de los valores se limita a un punto de supervisión específico identificado en la operación.

A continuación se muestra la signatura de la operación **setSpecificSlidingWindowParameters**:

```
void setSpecificSlidingWindowParameters (
            in ManagedEntityIdType nEManagedEntityId,
            in ManagedEntityIdType monitoringPoint,
            in MonitoredParameterSeqType monitoredParameterList,
            in short totConsecutiveIntvls,
            in short persistenceMinimum,
            in boolean allowGlobalOverwrite)
            raises (UnknownNE, UnknownParameters, IntervalCountTooLarge,
            AccessDenied, UnknownManagedEntity, CommFailure, EquipmentFailure);
```

El parámetro de entrada **nEManagedEntityId** identifica el elemento de red gestionado por el sistema de gestión del proveedor. El parámetro de entrada **monitoringPoint** identifica el punto de supervisión específico para establecer los parámetros de la ventana deslizante correspondientes a un parámetro supervisado. El parámetro de entrada **monitoredParameterList** identifica los parámetros cuyos valores se han de supervisar, definidos en Q834Common::MonitoredParameter. El parámetro de entrada **totConsecutiveIntvls** identifica el valor del total de intervalos consecutivos supervisados. El parámetro de entrada **persistenceMinimum** identifica el número mínimo de TCA para el monitoredParameter. El parámetro **allowGlobalOverwrite** indica si los parámetros de la ventana deslizante específica pueden sustituirse en una invocación posterior de la operación **setSlidingWindowParameters**.

El valor devuelto es del tipo **void**.

9.8.1.3 getSpecificSlidingWindowParameters (*obtener parámetros de ventana deslizante específicos*)

Esta operación proporciona una lista de los puntos de supervisión y sus valores de ventana deslizante correspondientes a un parámetro especificado.

A continuación se muestra la signatura de la operación **getSpecificSlidingWindowParameters**:

```
SWPValueSeqType getSpecificSlidingWindowParameters (
            in ManagedEntityIdType nEManagedEntityId,
            in MonitoredParameterType monitoredParameter)
            raises (UnknownNE, UnknownParameters, CommFailure);
```

El parámetro de entrada **nEManagedEntityId** identifica unívocamente el elemento de red gestionado por el sistema de gestión del proveedor. El parámetro de entrada **monitoredParameter** identifica el parámetro a supervisar y se define en q834_4::Q834Common::MonitoredParameter.

El valor devuelto es del tipo **SWPValueSeqType** y proporciona las asignaciones de ventana deslizante para un punto de supervisión específico.

9.8.1.4 setThreshold (*establecer umbral*)

Esta operación establece el valor del umbral identificado por el perfil de un punto de supervisión en un elemento de red.

A continuación se muestra la signatura de la operación **setThreshold**:

```
void setThreshold (
            in ManagedEntityIdType nEManagedEntityId,
            in ManagedEntityIdType monitoringPoint,
            in NameType thresholdDataProfileName )
                  raises (UnknownNE, AccessDenied,UnknownManagedEntity,
                  UnknownProfiles, InvalidAssociation,
                  CommFailure,ProfileSuspended);
```

El parámetro de entrada **nEManagedEntityId** identifica unívocamente el elemento de red gestionado por el sistema de gestión del proveedor. El parámetro de entrada **monitoringPoint** identifica el punto de supervisión específico para establecer los parámetros de ventana deslizante correspondientes a un parámetro supervisado. El parámetro de entrada **thresholdDataProfileName** proporciona una lista de nombres de perfil de los valores de umbral.

El valor devuelto es del tipo **void**.

9.8.1.5 setThresholds (*establecer umbrales*)

Esta operación permite establecer un conjunto de valores umbral, pares de monitoringPointType, en un NE concreto. Se plantea la excepción CommFailure si el sistema de gestión del proveedor es incapaz de comunicarse con el NE identificado en la operación y si los valores umbral se establecen directamente a partir de los recursos de la red.

Esta operación se utiliza para establecer valores por defecto para todo el sistema si el indicador sysScopeInd tiene el valor TRUE y si nEManagedEntityId es el identificador de un OLT. Esto significa que los valores de umbral han de aplicarse a todos los puntos de supervisión de cualquier componente del equipo que pertenezca actualmente, o que pueda pertenecer, al sistema que tiene la cabecera del OLT nombrado. Si el OLT hubiera perdido la comunicación con un recurso de red subyacente antes de la invocación de esta operación, o durante la misma, corresponde a la implementación del proveedor garantizar la aplicación de estos valores una vez restablecida la comunicación. Si el indicador sysScopeInd tuviera el valor TRUE y el nEManagedEntityId fuera el identificador de una ONU, los valores se aplicarían a la ONU y a todos los NT subyacentes. Conforme se añaden componentes de equipo al sistema que tiene la cabecera del OLT o de la ONU, el sistema de gestión del proveedor establecerá automáticamente estos valores umbral.

Si sysScopeInd tiene el valor FALSE, los valores umbral se aplicarán únicamente al recurso indicado. Estos valores se seguirán aplicando, no obstante, a los componentes del equipo, existentes o posibles, del nodo.

A continuación se muestra la signatura de la operación **setThresholds**:

```
void setThresholds (
                    in ManagedEntityIdType nEManagedEntityId,
                    in boolean sysScopeInd,
                    in ThresholdsSeqType thresholdsList)
                    raises (UnknownNE, UnknownProfiles,
                    AccessDenied,UnknownMonitoringPointTypes,
                    InvalidAssociation, CommFailure, ProfileSuspended );
```

El parámetro de entrada **nEManagedEntityId** identifica unívocamente el elemento de red gestionado por el sistema de gestión del proveedor. El parámetro de entrada **thresholdList** identifica una lista de pares consistentes en un nombre de datos de umbral y un monitoringPointType. La entrada **sysScopeInd** indica si los valores de la ventana deslizante deben aplicarse, o no, a todos los elementos de red subyacentes a partir del inicial designado en la operación.

El valor devuelto es del tipo **void**.

9.8.1.6 getThresholdValues (*obtener valores umbral*)

Esta operación proporciona un listado de puntos de supervisión y de sus correspondientes nombres de valores de datos umbral.

A continuación se muestra la signatura de la operación **getThresholdValues**:

```
MonitoringPointThresholdsSeqType getThresholdValues (
          in ManagedEntityIdType nEManagedEntityId,
          in MonitoringKindType monitoringPointType)
          raises (UnknownNE, UnknownMonitoringPointTypes, CommFailure);
```

El parámetro de entrada **nEManagedEntityId** identifica unívocamente el elemento de red gestionado por el sistema de gestión del proveedor. El parámetro de entrada **monitoringPointType** identifica el punto de supervisión sobre el que se basa la recuperación.

El valor devuelto es del tipo **MonitoringPointThresholdsSeqType** (***tipo de secuencia de umbrales de punto de supervisión***) y proporciona los valores de cruce de los umbrales.

9.8.1.7 getSystemThresholdsSetting (*obtener valores umbral del sistema*)

Esta operación recupera los valores por defecto del sistema correspondientes a los datos umbral.

A continuación se indica la signatura de la operación **getSystemThresholdsSetting**:

```
ThresholdsSeqType getSystemThresholdsSetting (
                  in ManagedEntityIdType nEManagedEntityId)
                  raises (AccessDenied, UnknownManagedEntity);
```

El parámetro de entrada **nEManagedEntityId** identifica unívocamente el sistema para el que se solicitan los valores por defecto.

El valor devuelto es del tipo **ThresholdsSeqType** y proporciona la lista de valores por defecto del sistema.

9.8.1.8 getSystemSWSettings (*obtener valores de la ventana deslizante del sistema*)

Esta operación recupera los valores de la ventana de deslizamiento por defecto del sistema.

A continuación se indica la signatura de la operación **getSystemSWSettings**:

```
ParameterSettingSeqType getSystemSWSettings (
                  in ManagedEntityIdType nEManagedEntityId)
                  raises (AccessDenied, UnknownManagedEntity);
```

El parámetro de entrada **nEManagedEntityId** identifica unívocamente el sistema para el que se solicitan los valores de la ventana deslizante por defecto.

El valor devuelto es del tipo **ParameterSettingSeqType** y proporciona la lista de valores por defecto del sistema.

9.8.1.9 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema de gestión del operador no tiene autorización de acceso a este objeto de interfaz.

La excepción **CommFailure** (***fallo de comunicaciones***) se plantea cuando falla el enlace RCD entre el NE y el sistema de gestión del proveedor.

La excepción **EquipmentFailure** (***fallo del equipo***) se plantea cuando el equipo está averiado y no pueden aplicarse los valores.

La excepción **InvalidAssociation** (***asociación no válida***) se plantea cuando el perfil indicado no puede aplicarse a un punto de supervisión.

La excepción **IntervalCountTooLarge** (***cómputo del intervalo demasiado grande***) se plantea cuando el intervalo solicitado supera el máximo soportado por el sistema de gestión del proveedor. Esta excepción indica el máximo número de intervalos de supervisión permitidos en el sistema de gestión del proveedor.

La excepción **ProfileSuspended** (***perfil suspendido***) se plantea cuando el OMS o el operador han suspendido el uso en el sistema de gestión del proveedor de los perfiles nombrados en la invocación.

La excepción **UnknownNE** (***NE desconocido***) se plantea cuando el NE mencionado en la solicitud es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownManagedEntity** (***entidad gestionada desconocida***) se plantea cuando el punto de supervisión es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownMonitoringPointTypes** (***tipos de punto de supervisión desconocidos***) se plantea cuando el punto de supervisión es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownParameters** (***parámetros desconocidos***) se plantea cuando el parámetro supervisado en cuestión es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownProfiles** (***perfiles desconocidos***) se plantea cuando el nombre de perfil proporcionado es desconocido para el sistema de gestión del proveedor y no puede recuperarse del repositorio de objetos de perfil.

9.8.2 Interfaz ReportController (*controlador de informes*)

9.8.2.1 addCustomerMonitoringReporting (*añadir informe de supervisión de cliente*)

Esta operación añade un historydatatype (*tipo de datos histórico*) a considerar en un punto de supervisión específico como respuesta a una queja de cliente identificada por el serviceInstanceId suministrado o para soportar servicios CNM.

A continuación se muestra la operación **addCustomerMonitoringReporting**:

```
void addCustomerMonitoringReporting  (
          in ManagedEntityIdType nEManagedEntityId,
          in ServiceInstanceIdType serviceInstanceId,
          in ManagedEntityIdType monitoringPoint,
          in GeneralizedTimeType stopTime,
          in HistoryDataType historyData,
          in short granularityPeriod)
          raises ( UnknownServiceInstance, AccessDenied, UnknownNE,
          UnknownManagedEntity, CollectionPeriodPast,
          CollectionLimitation, InvalidAssociation, UnknownHistoryDataType,
          CommFailure);
```

El parámetro de entrada **nEManagedEntityId** identifica unívocamente el elemento de red gestionado por el sistema de gestión del proveedor. El parámetro de entrada **serviceInstanceId** identifica el ejemplar de servicio de cliente asociado al punto de supervisión. El parámetro de entrada **monitoringPoint** identifica el punto de supervisión específico para la recogida de datos históricos. El parámetro de entrada **stopTime** identifica la hora de detención de la recogida de datos históricos. Cuando los informes deban continuar, se indicará stopTime igual a "0". Asimismo los informes terminarán automáticamente cuando se suprima el servicio. El parámetro de entrada **historyData** identifica el tipo de datos históricos a recoger, cuyos valores se definen en q834_4::Q834Common::RecordSetType. El parámetro de entrada **granularityPeriod** identifica el intervalo de recogida en minutos. Cuando **stopTime** está dentro del periodo de granularidad, los datos históricos se recogen durante todo el periodo de granularidad.

El valor devuelto es del tipo **void**.

9.8.2.2 removeCustomerMonitoringReporting (*suprimir informes de supervisión de cliente*)

Esta operación suprime toda la recogida de tipos de datos históricos de un ejemplar de servicio suministrado.

A continuación se muestra la signatura de la operación **removeCustomerMonitoringReporting**:

```
void removeCustomerMonitoringReporting  (
              in ServiceInstanceIdType serviceInstanceId )
              raises ( UnknownServiceInstance, AccessDenied,
              CollectionPeriodPast, CommFailure);
```

El parámetro de entrada **serviceInstanceId** identifica el ejemplar de servicio de cliente asociado al punto de supervisión.

El valor devuelto es del tipo **void**.

9.8.2.3 selectByServiceInstance (*seleccionar por ejemplar de servicio*)

Esta operación obtiene todos los registros disponibles en el sistema de gestión del proveedor asociados al serviceInstanceId en cuestión. La lista devuelta no tiene registros duplicados. El sistema de gestión del proveedor explora todos los conjuntos de registros pertinentes.

A continuación se muestra la signatura de la operación **selectByServiceInstance**:

```
RecordsSeqType selectByServiceInstance (
              In ServiceInstanceIdType serviceInstanceId,
              in GeneralizedTimeType intervalStartTime,
              in GeneralizedTimeType intervalEndTime)
              raises ( UnknownServiceInstance, AccessDenied);
```

El parámetro de entrada **serviceInstanceId** identifica el ejemplar de servicio del cliente. El parámetro de entrada **intervalStartTime** efectúa una selección por filtrado del registro de datos históricos con periodEndTime (*hora final del periodo*) anterior a este valor. El parámetro de entrada **intervalEndTime** (***hora final del intervalo***) excluye los registros de datos históricos con periodEndTime posterior a este valor.

El valor devuelto es del tipo **RecordsSeqType** y proporciona una lista de registros de datos históricos asociados a la supervisión de la calidad de funcionamiento comunicada por este ejemplar de servicio.

9.8.2.4 displayActiveReporting (*mostrar informes activos*)

Esta operación obtiene todos los puntos de supervisión para los que se están recogiendo datos históricos correspondientes a un determinado serviceInstanceId.

A continuación se muestra la signatura de la operación **displayActiveReporting**:

```
MonitoringPointSeqType displayActiveReporting (
              In ServiceInstanceIdType serviceInstanceId)
              raises ( UnknownServiceInstance, AccessDenied);
```

El parámetro de entrada **serviceInstanceId** identifica el ejemplar de servicio del cliente.

El valor devuelto es del tipo **MonitoringPointSeqType** (*tipo de secuencia de punto de supervisión*) y proporciona el listado de todos los puntos de supervisión que están informando.

9.8.2.5 addNewMonitoringReporting (*añadir nuevo informe de supervisión*)

Esta operación añade un tipo de datos históricos a supervisar en un punto de supervisión específico.

A continuación se muestra la signatura de la operación **addNewMonitoringReporting**:

```
void addNewMonitoringReporting  (
              in ManagedEntityIdType nEManagedEntityId,
              in ManagedEntityIdType monitoringPoint,
              in GeneralizedTimeType stopTime,
              in HistoryDataType historyData,
              in short granularityPeriod)
              raises ( AccessDenied, UnknownNE,
              UnknownManagedEntity, CollectionPeriodPast,
              CollectionLimitation, InvalidAssociation, UnknownHistoryDataType,
              CommFailure);
```

El parámetro de entrada **nEManagedEntityId** identifica unívocamente el elemento de red gestionado por el sistema de gestión del proveedor. El parámetro de entrada **monitoringPoint** identifica el punto de supervisión específico para la recogida de datos históricos. El parámetro de entrada **stopTime** identifica la hora a la que terminará la recogida de datos históricos. Si los informes han de seguir generándose, se indica stopTime igual a "0". Asimismo los informes terminan automáticamente al suprimir la managedEntity (punto de supervisión). El parámetro de entrada **historyData** identifica el tipo de datos históricos a recoger, cuyos valores se definen en q834_4::Q834Common::RecordSetType. El parámetro de entrada **granularityPeriod** identifica el periodo del intervalo de recogida en minutos. Cuando **stopTime** cae dentro del periodo de granularidad, los datos históricos se recogen durante todo el periodo de granularidad.

El valor devuelto es del tipo **void**.

9.8.2.6 selectByMonitoringPoint (*seleccionar por punto de supervisión*)

Esta operación obtiene todos los registros disponibles en el sistema de gestión del proveedor asociados al punto de supervisión en cuestión. La lista devuelta no tiene registros duplicados.

A continuación se muestra la signatura de la operación **selectByMonitoringPoint**:

```
RecordsSeqType selectByMonitoringPoint (
              in ManagedEntityIdType monitoringPoint,
              in GeneralizedTimeType intervalStartTime,
              in GeneralizedTimeType intervalEndTime)
              raises ( UnknownManagedEntity, AccessDenied);
```

El parámetro de entrada **monitoringPoint** identifica la supervisión específica cuyos registros deben recuperarse. El parámetro de entrada **intervalStartTime** (***hora de comienzo del intervalo***) excluye los registros de datos históricos cuyo periodEndTime es anterior a este valor. El parámetro de entrada **intervalEndTime** (***hora de finalización del intervalo***) permite ignorar los registros de datos históricos cuyos periodEndTime es posterior a este valor.

El valor devuelto es del tipo **RecordsSeqType** y proporciona todos los registros de datos históricos almacenados en el sistema de gestión del proveedor por el punto de supervisión durante el periodo de tiempo especificado.

9.8.2.7 createReportingSchedule (*crear planificación de informes*)

Esta operación añade una recogida de datos históricos planificada a supervisar en un punto de supervisión específico. El sistema de gestión del proveedor recogerá un registro de datos históricos del punto de supervisión con arreglo a la planificación. La planificación identifica los periodEndTime para los registros que han de recogerse.

A continuación se muestra la signatura de la operación **createReportingSchedule**:

```
void createReportingSchedule  (
              in ManagedEntityIdType nEManagedEntityId,
              in ManagedEntityIdType monitoringPoint,
              in HistoryDataType historyData,
              in ServiceInstanceIdType serviceInstance,
              in short granularityPeriod,
              in UserLabelType schedulerName)
              raises (AccessDenied, UnknownNE,
              UnknownManagedEntity, CollectionLimitation, UnknownScheduler,
              InvalidAssociation, UnknownHistoryDataType, InvalidScheduler);
```

El parámetro de entrada **nEManagedEntityId** identifica unívocamente el elemento de red gestionado por el sistema de gestión del proveedor. El parámetro de entrada **monitoringPoint** identifica el punto de supervisión específico para la recogida de datos históricos. El parámetro de entrada **historyData** identifica el tipo de datos históricos a recoger, cuyos valores se definen en q834_4::Q834Common::RecordSetType. El parámetro de entrada **serviceInstance** proporciona una referencia opcional a un ejemplar de servicio. Se utiliza el valor cadena de caracteres vacía cuando no se desea ninguna referencia. El parámetro de entrada **granularityPeriod** identifica el periodo de recogida en minutos. El parámetro de entrada **schedulerName** identifica el nombre del planificador creado en algún instante anterior.

El valor devuelto es del tipo **void**.

9.8.2.8 modifyReportingSchedule (*modificar planificación de informes*)

Esta operación modifica la recogida de datos históricos planificada a supervisar en un punto de supervisión específico. Cuando se completa con éxito, la modificación de la recogida de datos históricos planificada tiene lugar en la siguiente iteración.

A continuación se muestra la signatura de la operación **modifyReportingSchedule**:

```
void modifyReportingSchedule  (
               in ManagedEntityIdType nEManagedEntityId,
               in ManagedEntityIdType monitoringPoint,
               in HistoryDataType historyData,
               in UserLabelType newSchedulerName)
               raises (AccessDenied, UnknownNE,
               UnknownManagedEntity, CollectionLimitation, UnknownScheduler,
               InvalidAssociation, UnknownHistoryDataType, InvalidScheduler);
```

El parámetro de entrada **nEManagedEntityId** identifica unívocamente el elemento de red gestionado por el sistema de gestión del proveedor. El parámetro de entrada **monitoringPoint** identifica el punto de supervisión específico para la recogida de datos históricos. El parámetro de entrada **historyData** identifica el tipo de datos históricos a recoger, cuyos valores se definen en q834_4::Q834Common::RecordSetType. El parámetro de entrada **newSchedulerName** identifica el nombre de la nueva planificación a utilizar.

El valor devuelto es del tipo **void**.

9.8.2.9 cancelReportingSchedule (*cancelar planificación de informes*)

Esta operación cancela la generación posterior de informes planificados de datos históricos solicitados anteriormente por el operador. No se interrumpe la generación de informes de datos históricos en curso ni la de informes soportada como comportamiento por defecto entre el sistema de gestión del proveedor y los recursos de red.

A continuación se muestra la signatura de la operación **cancelReportingSchedule**:

```
void cancelReportingSchedule (
               in ManagedEntityIdType nEManagedEntityId,
               in ManagedEntityIdType monitoringPoint,
               in HistoryDataType historyData,
               in UserLabelType schedulerName)
               raises (AccessDenied, UnknownNE,
               UnknownManagedEntity, UnknownScheduler,
               InvalidAssociation, UnknownHistoryDataType);
```

El parámetro de entrada **nEManagedEntityId** identifica unívocamente el elemento de red gestionado por el sistema de gestión del proveedor. El parámetro de entrada **monitoringPoint** identifica el punto de supervisión específico para la recogida de datos históricos. El parámetro de entrada **historyData** identifica el tipo de datos históricos a recoger, cuyos valores se definen en q834_4::Q834Common::RecordSetType. El parámetro de entrada **schedulerName** identifica la planificación a cancelar.

El valor devuelto es del tipo **void**.

9.8.2.10 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema de gestión del operador no tiene autorización de acceso a este objeto de interfaz.

La excepción **CommFailure** (***fallo de comunicaciones***) se recibe cuando hay un fallo del enlace RCD entre el NE y el sistema de gestión del proveedor.

La excepción **CollectionLimitation** (***recogida limitada***) se plantea cuando el sistema de gestión del proveedor no puede recoger datos durante el tiempo y periodo de granularidad especificados debido a restricciones de la implementación.

La excepción **CollectionPeriodPast** (***plazo de recogida superado***) se plantea cuando la hora de terminación del plazo es inferior o igual a la hora actual.

La excepción **InvalidAssociation** (***asociación no válida***) se plantea cuando el perfil en cuestión no puede aplicarse a un punto de supervisión.

La excepción **InvalidScheduler** (***planificador no válido***) se plantea cuando el planificador en cuestión es inadecuado para esta operación o es obsoleto.

La excepción **UnknownNE** (***NE desconocido***) se plantea cuando el NE mencionado en la petición es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownHistoryDataType** (***tipo de datos histórico desconocido***) se plantea cuando el tipo de datos históricos es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownManagedEntity** (***entidad de gestión desconocida***) se plantea cuando el punto de supervisión es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownScheduler** (***planificador desconocido***) se plantea cuando el planificador nombrado es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownServiceInstance** (***ejemplar de servicio desconocido***) se plantea cuando el ejemplar de servicio es desconocido para el sistema de gestión del proveedor.

9.9 ProfileManager (*gestor de perfiles*)

Este módulo consta de tres interfaces: ProfileConsumer (*consumidor de perfiles*), ProfileUsageMgr (*gestor de uso de perfiles*) y ProfileRetriever (*recuperador de perfiles*). ProfileConsumer especifica el contenido de la notificación de eventos para la creación de perfiles en el repositorio de objetos de perfil. ProfileUsageMgr soporta las operaciones destinadas a gestionar los valores de los perfiles almacenados provisionalmente en el sistema de gestión del proveedor. ProfileRetriever permite al sistema de gestión del proveedor obtener los valores de los perfiles.

9.9.1 Interfaz ProfileConsumer (*consumidor de perfiles*)

El objeto de esta interfaz es anunciar la existencia de nuevos valores de perfil y comunicarlos al sistema de gestión del proveedor. Esta interfaz no tiene operaciones. No obstante proporciona la correspondencia del encabezamiento fijo así como las correspondencias de datos filtrables para el objeto de evento estructurado utilizado para entregar la información de evento por el canal de eventos del servicio de notificación del OMG.

En el encabezamiento fijo, se le da a **domain_type** el valor "telecommunications", a **type_name** se le da el valor "ProfileEvent" y a **event_name** se le da el valor de una cadena de caracteres constante cuyo valor es "ProfileCreation" definido en la interfaz.

La correspondencia de los datos filtrables emplea una estrategia que utiliza un identificador de cadena de caracteres constante para el componente de datos filtrables seguido del valor de dicho elemento de datos. Además, los elementos de datos filtrables se listan en un orden específico.

El orden de los elementos filtrables es el siguiente: ProfileName, ProfileType, EventTime, ProfileAttributeValues y NotificationIdentifier. El valor de ProfileName tiene la sintaxis de NameType. El valor de ProfileType tiene la sintaxis "unsigned short" (*precisión corta sin signo*), y permite al consumidor (sistema de gestión del proveedor) identificar el tipo de perfil creado y desordenar los valores de los atributos que aparecen más adelante en profileStruct. La sintaxis de ProfileAttributeValues es profileStruct. EventTime tiene la sintaxis de GeneralizedTimeType y es la hora de creación del perfil. NotificationIdentifier tiene la sintaxis de NotificationIdentifierType. El identificador etiqueta unívocamente el evento creación de perfil y se incrementa cada vez que se crea un nuevo perfil.

9.9.2 Interfaz ProfileUsageMgr (*gestor de uso de perfiles*)

9.9.2.1 reName (*modificar nombre*)

Esta operación permite modificar el nombre de un perfil.

A continuación se muestra la signatura de la operación **reName**:

```
void reName (
            in NameType  oldProfileName,
            in NameType newProfileName)
            raises (UnknownProfiles, AccessDenied, DuplicateProfileName);
```

El parámetro de entrada **oldProfileName** es el antiguo nombre del perfil cuyo nombre se ha de modificar. El parámetro de entrada **newProfileName** es el nuevo nombre del perfil.

El valor devuelto es del tipo **void**.

9.9.2.2 inUse (*utilizándose*)

Esta operación devuelve un valor booleano que indica si el perfil está utilizándose.

A continuación se muestra la signatura de la operación **inUse**:

```
boolean inUse (
            in NameType profileName)
            raises (UnknownProfiles, AccessDenied);
```

El parámetro de entrada **profileName** proporciona el nombre del perfil que ha de verificarse para ver si está siendo utilizado por un tercero.

El valor devuelto es del tipo **boolean**.

9.9.2.3 suspendUse (*suspender uso*)

Esta operación suspende el uso de un perfil.

A continuación se muestra la signatura de la operación **suspendUse**:

```
void suspendUse (
              in NameType profileName)
              raises (UnknownProfiles, AccessDenied);
```

El parámetro de entrada **profileName** es el nombre del perfil que ha de suspenderse.

El valor devuelto es del tipo **void**.

9.9.2.4 resumeUse (*reanudar uso*)

Esta operación reanuda el uso del perfil nombrado.

A continuación se muestra la signatura de la operación **resumeUse**:

```
void resumeUse (
          in NameType profileName)
          raises (UnknownProfiles, AccessDenied) ;
```

El parámetro de entrada **profileName** nombra el perfil cuyos valores han de estar disponibles de nuevo para que los utilice el sistema de gestión del proveedor.

El valor devuelto es del tipo **void**.

9.9.2.5 deleteProfile (*suprimir perfil*)

Esta operación permite suprimir un perfil que no se esté utilizando.

A continuación se indica la signatura de la operación **deleteProfile**:

```
void deleteProfile (in NameType profileName) raises (UnknownProfiles,
AccessDenied, ProfileInUse);
```

El parámetro de entrada **profileName** es el nombre del perfil a suprimir.

El valor devuelto es del tipo **void**.

9.9.2.6 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el OMS no tiene autorización de acceso a la operación solicitada.

La excepción **DuplicateProfileName** (***nombre de perfil duplicado***) se plantea cuando se encuentra un nombre de perfil duplicado.

La excepción **ProfileInUse** (***perfil utilizándose***) se plantea al intentar suprimir un perfil que está siendo utilizado por un tercero.

La excepción **UnknownProfiles** (***perfiles desconocidos***) se plantea cuando el nombre del perfil proporcionado es desconocido para el sistema de gestión del proveedor y no puede recuperarse del repositorio de objetos de perfil.

9.9.3 Interfaz ProfileRetriever (*recuperador de perfiles*)

9.9.3.1 retrieve (*recuperar*)

Esta operación permite recuperar un perfil. Esta operación la utiliza el sistema de gestión del proveedor para recuperar un perfil del sistema de gestión del operador.

A continuación se muestra la signatura de la operación **retrieve**:

```
ProfileInfoType retrieve (
              in NameType profileName)
              raises (UnknownProfiles);
```

El parámetro de entrada **profileName** es el nombre del perfil a recuperar.

El valor devuelto es del tipo **ProfileInfoType** y proporciona los valores de atributo para el perfil nombrado.

9.9.3.2 Excepciones

La excepción **UnknownProfiles** (***perfiles desconocidos***) se plantea cuando el nombre del perfil proporcionado es desconocido para el sistema de gestión del proveedor y no puede recuperarse del repositorio de objetos de perfil.

9.10 Módulo Registrar (*registro*)

Este módulo soporta el proceso de pasar un nuevo NE a la jurisdicción de gestión del sistema de gestión del proveedor. Soporta tanto las instalaciones iniciales y las sustituciones de equipos por motivos de mantenimiento o actualizaciones de servicio. Soporta explícitamente la función de determinación de distancia utilizada para la medición del retardo de ida y vuelta entre el OLT y cada ONU u ONT y para establecer los mecanismos de seguridad (algoritmo de batido de claves) y temporización de la comunicación incluido el establecimiento automático del canal de operaciones integradas.

Como resultado del registro se crea automáticamente un conjunto de entidades gestionadas en el modelo de información de gestión mantenido por el sistema de gestión del proveedor. Dependiendo del tipo de equipos, se crean automáticamente ejemplares de OLT, ONT, ONU, APONLink, APONTTP, APONTrail, APONNetworkCTP, APONNetworkTTP, APONLinkConnection, tcAdapterF, vpLinkConnectionF, vpTopologicalLinkF y physicalPathTPF. Dependiendo de la

implementación del proveedor y de la instalación física, se crean también automáticamente otras entidades gestionadas.

9.10.1 Interfaz NERegistrar (*registro del NE*)

9.10.1.1 registerNE (*registrar NE*)

Esta operación registra un NE en un ejemplar concreto del sistema de gestión del proveedor. El NE debe estar instalado en este punto. Se devuelve el Id de entidad gestionada del NE para indicar que el sistema de gestión del proveedor puede ejecutar como mínimo una función de gestión rudimentaria.

A continuación se muestra la signatura de la operación **registerNE**:

```
ManagedEntityIdType registerNE (
                    in DCNAddressType nEDCNAddress,
                    in UserLabelType nEUserLabel,
                    in AdministrationDomainType administrationDomain)
                    raises (AccessDenied, DCNTimeout, AddressLabelMismatch,
                    DuplicateUserLabel, TooManyNEs, InvalidDCNAddress,
                    DeniedAccess, InvalidUserLabelSyntax);
```

El parámetro de entrada **nEDCNAddress** identifica la dirección RCD a utilizar para acceder a este NE. El NE ya debe estar configurado con esta dirección. El parámetro de entrada **nEUserLabel** proporciona una etiqueta definida por el operador para el NE. El NE ya debe haber sido dotado de esta etiqueta de usuario. El parámetro de entrada **administrationDomain** identifica el dominio de gestión al que está asignado este NE.

El valor devuelto es del tipo **ManagedEntityIdType** e identifica el nuevo NE registrado cuando esta operación se completa con éxito.

9.10.1.2 modifyNEDCNAddress (*modificar dirección RCD del NE*)

Esta operación modifica la dirección RCD de un NE registrado. Durante esta operación puede perderse temporalmente la comunicación entre el sistema de gestión del proveedor y el NE. La nueva dirección RCD debe tener efecto inmediatamente.

A continuación se muestra la signatura de la operación **modifyNEDCNAddress**:

```
void modifyNEDCNAddress (
       in ManagedEntityIdType nEManagedEntityId,
       in DCNAddressType newNEDCNAddress)
       raises(AccessDenied, DeniedAccess, AddressLabelMismatch,
       DCNTimeout, CommFailure, UnknownNE, InvalidDCNAddress, BackupInProgress);
```

El parámetro de entrada **nEManagedEntityId** identifica el NE al que se aplica esta operación. El parámetro de entrada **newDCNAddress** identifica la nueva dirección RCD del NE.

El valor devuelto es del tipo **void**.

9.10.1.3 rangeONTorONU (*añadir ONT u ONU*)

Esta operación añade un ONT u ONU utilizando el número de serie de la interfaz PON del NE. Si el NE que se añade no se hubiera construido, se llamará a buildNode antes de que ocurra la adición. Una vez completada la adición con éxito se inicia el synch de NE. Esto actualizará el sistema de gestión del proveedor con los datos de configuración del nuevo NE añadido.

A continuación se muestra la signatura de la operación **rangeONTorONU**:

```
ManagedEntityIdType rangeONTorONU(
          in ManagedEntityIdType oLTManagedEntityId,
          in UserLabelType nEUserLabel,
          in SerialNumType serialNum,
```

```
        in ManagedEntityIdType port)
        raises(AccessDenied, CommFailure, EquipmentFailure,
        UnknownNE, UnknownPort, MaxSubtendingNodesExceeded, InsufficientPONBW,
        InvalidSerialNumSyntax, APONLayerFailure, DuplicateUserLabel,
        InvalidUserLabelSyntax, BackupInProgress, SynchInProgress );
```

El parámetro de entrada **oLTManagedEntityId** identifica el OLT al que se aplica esta operación. El parámetro de entrada **nEUserLabel** es la etiqueta asignada al NE que se añade. El parámetro de entrada **serialNum** identifica el número de serie del NE. El parámetro de entrada **port** identifica el puerto PON específico del OLT al que se añade el NE.

El valor devuelto es del tipo **ManagedEntityIdType** y es el id de entidad gestionada del NE añadido.

9.10.1.4 rangeReplacementNE (*añadir NE de repuesto*)

Esta operación añade un ONT u ONU de repuesto utilizando el número de serie del equipo de repuesto. Toda la información de conexión del servicio existente se asigna automáticamente al NE de repuesto. Los casos de repuesto corresponden a las actividades de mantenimiento destinadas a los clientes y servicios existentes. La nueva etiqueta de usuario del NE y la vieja pueden coincidir. Fuera del ámbito de IF1 el proveedor y el operador acuerdan qué tipo de soporte físico puede sustituir a otro existente.

A continuación se muestra la signatura de la operación **rangeReplacementNE**:

```
ManagedEntityIdType rangeReplacementNE(
              in ManagedEntityIdType oldNEManagedEntityId,
              in UserLabelType newNEUserLabel,
              in SerialNumType replacementSerialNum)
              raises (AccessDenied, CommFailure, UnknownNE,
              InvalidSerialNumSyntax, APONLayerFailure, EquipmentFailure,
              InvalidUserLabelSyntax, HWServicesMismatch,
              DuplicateUserLabel, BackupInProgress, SynchInProgress );
```

El parámetro de entrada **oldNEManagedEntityId** identifica el NE a sustituir. El parámetro de entrada **newNEUserLabel** identifica la etiqueta para el nuevo ONT u ONU que se añade. El parámetro de entrada **replacementSerialNum** identifica el número de serie del NE.

El valor devuelto es del tipo **ManagedEntityIdType** y es el id de entidad gestionada del nuevo NE añadido.

9.10.1.5 rangeUpgradeNE (*añadir actualización de NE*)

Esta operación añade un NE de repuesto para actualizar el soporte físico. Toda la información de conexión del servicio existente se asigna automáticamente al NE de repuesto. Además, se supone que el NE de repuesto se ha dotado previamente (por medio de la operación buildNode) y que se han suministrado nuevas conexiones de servicio o se ha reservado anchura de banda. La nueva etiqueta de usuario del NE y la antigua pueden coincidir. Fuera el ámbito de IF1, el proveedor y el operador acuerdan qué tipo de soporte físico puede sustituir a un soporte físico existente.

A continuación se muestra la signatura de la operación **rangeUpgradeNE**:

```
ManagedEntityIdType rangeUpgradeNE(
              in ManagedEntityIdType oldNEManagedEntityId,
              in ManagedEntityIdType newNEManagedEntityId,
              in UserLabelType newNEUserLabel,
              in SerialNumType newNESerialNum )
              raises (AccessDenied, CommFailure,
              APONLayerFailure, EquipmentFailure, InvalidUserLabelSyntax,
              DuplicateUserLabel, UnknownNE,
              HWServicesMismatch, InsufficientPONBW, BackupInProgress,
              SynchInProgress );
```

El parámetro de entrada **oldNEManagedEntityId** identifica el NE a sustituir. En este caso el tipo de NE que se sustituye es o bien un ONT o bien una ONU. El parámetro de entrada **newNEManagedEntityId** identifica la sustitución suministrada previamente. Toda la información de nueva conexión de servicio o anchura de banda es accesible al sistema de gestión del proveedor a través de este identificador. El parámetro de entrada **newNEUserLabel** proporciona una etiqueta para el nuevo NE a añadir. La nueva etiqueta de usuario de la ONU u ONT pueden coincidir. El parámetro de entrada **newNESerialNum** proporciona el número de serie a utilizar para añadir el repuesto.

El valor devuelto es del tipo **ManagedEntityIdType** y es el ID de entidad gestionada del nuevo NE añadido.

9.10.1.6 moveONTorONU (*trasladar ONT u ONU*)

Esta operación se utiliza para trasladar un ONT u ONU de una PON a otra y también para trasladar todos los servicios asociados.

A continuación se muestra la signatura de la operación **moveONTorONU**:

```
ManagedEntityIdType moveONTorONU(
                    in ManagedEntityIdType oldNEManagedEntityId,
                    in ManagedEntityIdType newPONPort)
                    raises(AccessDenied, CommFailure, UnknownNE, UnknownPort,
                    APONLayerFailure,EquipmentFailure, InsufficientPONBW,
                    BackupInProgress, SynchInProgress );
```

El parámetro de entrada **oldNEManagedEntityId** identifica el NE a sustituir. El parámetro de entrada **newPONPort** identifica la nueva PON a la que se trasladan estos servicios.

El valor devuelto es del tipo **ManagedEntityIdType** y es el ID de entidad gestionada del nuevo NE añadido

9.10.1.7 getSubtendingNEList (*obtener lista de NE subyacentes*)

Esta operación devuelve todos los NE subyacentes de un NE específico.

A continuación se muestra la signatura de la operación **getSubtendingNEList**:

```
ManagedEntityIdSeqType getSubtendingNEList(
                    in ManagedEntityIdType nEManagedEntityId)
                    raises (UnknownNE, AccessDenied);
```

El parámetro de entrada **nEManagedEntityId** identifica el NE específico para el que debe devolverse la lista.

El valor devuelto es del tipo **ManagedEntityIdSeqType** y proporciona una lista de los elementos de red subyacentes.

9.10.1.8 nEListGet (*obtener lista de NE*)

Esta operación recupera los elementos de red sometidos a la jurisdicción de gestión del sistema de gestión del proveedor.

A continuación se indica la signatura de la operación **nEListGet**:

```
ManagedEntityIdSeqType nEListGet () raises (AccessDenied);
```

No hay parámetros de entrada.

El valor devuelto es del tipo **ManagedEntityIdSeqType** y proporciona la lista de elementos de red gestionados por el sistema de gestión del proveedor.

9.10.1.9 deRegisterNE (*suprimir registro del NE*)

Esta operación suprime el elemento de red de la jurisdicción de gestión del sistema de gestión del proveedor.

A continuación se indica la signatura de la operación **deRegisterNE**:

```
void deRegisterNE (
            in ManagedEntityIdType nE)
            raises (AccessDenied);
```

El parámetro de entrada **nE** identifica el elemento de red a suprimir de la jurisdicción de gestión del sistema de gestión del proveedor.

El valor devuelto es del tipo **void.**

9.10.1.10 associateNE (*asociar NE*)

Esta operación asocia la información de suministro previo a un elemento de red que se haya instalado o descubierto automáticamente. Las actividades de suministro previo y de instalación producen ambas una entidad gestionada en el sistema de gestión del proveedor que se pone a disposición del OMS. Esta operación funde las dos entidades gestionadas en una sola.

A continuación se indica la signatura de la operación **associateNE**:

```
ManagedEntityIdType associateNE (
                  in ManagedEntityIdType preProvisionedNE,
                  in ManagedEntityIdType discoveredNE)
                  raises (AccessDenied, UnknownManagedEntity);
```

El parámetro de entrada **preProvisionedNE** identifica la información asociada al elemento de red suministrado previamente. El parámetro de entrada **discoveredNE** identifica la información asociada al elemento de red instalado.

El valor devuelto es del tipo **ManagedEntityIdType** y proporciona la identificación de la información unificada.

9.10.1.11 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema no tiene autorización de acceso al objeto de interfaz.

La excepción **AddressLabelMismatch** (***desadaptación de las etiquetas de dirección***) se plantea cuando el NE identificado no dispone de la dirección RCD actual proporcionada en la petición.

La excepción **APONLayerFailure** (***fallo de la capa APON***) se plantea cuando se presenta un fallo de determinación de protocolo APON entre el OLT y el nodo subyacente designado.

La excepción **BackupInProgress** (***copia de seguridad en curso***) se plantea cuando se emite la petición de cancelación habiendo una copia de seguridad en curso.

La excepción **CommFailure** (***fallo de comunicaciones***) se plantea cuando cae la RCD entre el sistema de gestión del proveedor y el OLT o se interrumpe la comunicación entre el OLT y el ONT de origen.

La excepción **DCNTimeout** (***fin de temporización de la RCD***) se plantea cuando el enlace de comunicaciones de la RCD entre un NE como mínimo y el sistema de gestión del proveedor está tan congestionado que el estado actual de la información de estado no puede transferirse dentro de un plazo synch definido por el sistema.

La excepción **DeniedAccess** (***denegación de acceso***) se plantea cuando se deniega el acceso a un NE como resultado de las restricciones de control de acceso.

La excepción **DuplicateUserLabel** (***etiqueta de usuario duplicada***) se plantea cuando la etiqueta de usuario proporcionada en la petición se ha utilizado para etiquetar otro NE o unidad enchufable. Dicho de otro modo, el sistema de gestión del proveedor se encarga de supervisar la asignación de etiquetas de usuarios a los NE y a las unidades enchufables en su jurisdicción de gestión.

La excepción **EquipmentFailure** (***fallo de equipo***) se plantea cuando el equipo de cuyos datos se hace copia de seguridad está averiado.

La excepción **HWServicesMismatch** (***desadaptación de los servicios de soporte físico***) se plantea cuando el NE de repuesto no puede ejecutar los servicios prestados.

La excepción **InsufficientPONBW** (***anchura de banda de PON insuficiente***) se plantea cuando el ONT o la ONU no pueden añadirse debido a que la anchura de banda del APONLink es insuficiente.

La excepción **InvalidDCNAddress** (***dirección RCD no válida***) se plantea cuando la dirección RCD especificada no es válida.

La excepción **InvalidSerialNumSyntax** (***sintaxis de número de serie no válida***) se plantea cuando la sintaxis del número de serie proporcionado no se ajusta a las reglas de definición.

La excepción **InvalidUserLabelSyntax** (***sintaxis de etiqueta de usuario no válida***) se plantea cuando la etiqueta de usuario proporcionada a la ONU o al ONT infringe las reglas de sintaxis de la empresa definidas por el operador e implementadas en el sistema de gestión del proveedor.

La excepción **MaxSubtendingNodesExceeded** (***sobrepasado el número máximo de nodos subyacentes***) se plantea cuando el número máximo de nodos subyacentes especificados para la interfaz PON identificada se ha sobrepasado con esta petición de prestación de servicio.

La excepción **SynchInProgress** (***sincronización en curso***) se plantea cuando se solicita una operación mientras el sistema de gestión del proveedor se encuentra en proceso de sincronización con el NE.

La excepción **TooManyNEs** (***demasiados NE***) se plantea cuando el sistema de gestión del proveedor no puede gestionar ninguna OLT más.

La excepción **UnknownManagedEntity** (***entidad gestionada desconocida***) se plantea cuando el equipo es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownNE** (***NE desconocido***) se plantea cuando el OLT es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownPort** (***puerto desconocido***) se plantea cuando el puerto identificado es desconocido para el sistema de gestión del proveedor.

9.11 Módulo ResourceAllocation (*asignación de recursos*)

Este servicio permite al OMS reservar anchura de banda de los recursos del sistema para una conexión anticipada de servicio. La anchura de banda reservada puede suprimirse o recuperarse. La interfaz ResourceAllocator proporciona los medios de crear, suprimir o mostrar las reservas de recursos. Esta operación se utiliza antes de enviar a los técnicos a instalar un elemento de red. Una vez reservada la capacidad de recursos, el recurso reservado sólo puede utilizarse para el servicio especificado en la reserva.

9.11.1 Interfaz ResourceAllocator (*asignadora de recursos*)

9.11.1.1 reserveForService (*reservar para servicio*)

Esta operación reserva anchura de banda para un recurso de red tal como ONT, ONU o NT cuya instalación esté pendiente. Esta operación se utiliza cuando el ONT, la ONU o el NT que sirve la anchura de banda asociada al OLT designado se suministra por primera vez. Cuando se completa la

operación, el valor devuelto **ReservationBandwidthType** proporciona un cómputo de la anchura de banda reservada al OMS.

A continuación se muestra la signatura de la operación **reserveForService**:

```
ReservationBandwidthType reserveForService(
                         in EndPointType endPointA,
                         in EndPointType endPointZ,
                         in NameSeqType networkCharacteristicsProfiles,
                         in ServiceInstanceIdType serviceInstanceId)
                         raises(UnknownNE, UnknownPort, UnknownProfiles,
                         InsufficientBW, MaxSubtendingNodesExceeded,
                         ConnectionCountExceeded, CommFailure, AccessDenied,
                         ProfileSuspended );
```

Los parámetros de entrada **endPointA** y **endPointZ** identifican los puntos extremos de la conexión de servicio que determinará los recursos necesarios para soportar una conexión de servicio anticipada. El parámetro de entrada **networkCharacteristicsProfiles** es una lista que consiste en lo siguiente: abTrafficDescripterProfile y baTrafficDescripterProfile (en este orden). El parámetro de entrada **serviceInstanceId** identifica el ejemplar de servicio asociado a esta anchura de banda reservada.

El valor devuelto es del tipo **ReservationBandwidthType** e incluye un Id de reserva y la cantidad de anchura de banda de recurso de red reservada por el sistema de gestión del proveedor. El Id de reserva puede utilizarlo el OMS durante la prestación del servicio o bien cancelar esta reserva.

9.11.1.2 cancelReservation (*cancelar reserva*)

Esta operación se utiliza para suprimir la reserva y liberar los recursos de la capacidad del sistema reservado.

A continuación se muestra la signatura de la operación **cancelReservation**:

```
AvailableSysBandwidthSeqType cancelReservation (
                             in ReservationIdType reservationId )
                             raises (UnknownReservationId, CommFailure,
                             AccessDenied);
```

El parámetro de entrada **reservationId** identifica la reserva existente asociada a la anchura de banda asignada.

El valor devuelto es del tipo **AvailableSysBandwidthSeqType** (***tipo de secuencia de anchura de banda del sistema disponible***) e indica la anchura de banda del OLT actualmente disponible tras la supresión de la anchura de banda reservada.

9.11.1.3 getReservationId (*obtener Id de la reserva*)

Esta operación se utiliza para mostrar el Id de reserva asociado al Id del ejemplar de servicio al que se le asigna la anchura de banda reservada.

A continuación se muestra la signatura de la operación **getReservationId**:

```
ReservationIdType getReservationId (
                  in ServiceInstanceIdType serviceInstanceId)
                  raises(UnknownServiceInstance, AccessDenied);
```

El parámetro de entrada **serviceInstanceId** se utiliza para identificar los ejemplares de servicio existentes asignados durante la reserva del recurso.

El valor devuelto es del tipo **ReservationIdType** y se asocia al Id de ejemplar de servicio en cuestión.

9.11.1.4 reportReservedResources (*informar de los recursos reservados*)

Esta operación la utiliza el OMS para mostrar la anchura de banda actualmente reservada en el NE de cabecera específico (en este caso el OLT).

A continuación se muestra la signatura de la operación **reportReservedResources**:

```
ReservedBandwidthSeqType reportReservedResources (
                            in ManagedEntityIdType nEManagedEntityId)
                            raises (UnknownNE, AccessDenied);
```

El parámetro de entrada **nEManagedEntityId** se utiliza para identificar el elemento de red para el que se solicita la reserva actual de anchura de banda.

El valor devuelto es del tipo **ReservedBandwidthSeqType** y tiene toda la información de la anchura de banda actualmente reservada para el elemento de red.

9.11.1.5 getReservations (*obtener reservas*)

Esta operación la utiliza el OMS para recuperar todas las reservas asociadas al NE en cuestión.

A continuación se muestra la signatura de la operación **getReservations**:

```
ReservationIdSeqType getReservations(
              in ManagedEntityIdType nEManagedEntityId)
              raises (UnknownNE, AccessDenied);
```

El parámetro de entrada **nEManagedEntityId** se utiliza para identificar el NE para el que se recupera la anchura de banda actualmente reservada.

El valor devuelto es del tipo **ReservationIdSeqType** y proporciona todos los Id de reserva actualmente asignados al OLT.

9.11.1.6 cancelAllRemainingReservations (*cancelar todas las reservas pendientes*)

Esta operación se utiliza para suprimir todas las reservas de capacidad pendientes asociadas a un elemento de red determinado. El elemento de red puede tener anchura de banda asignada, reservada o disponible para conexiones de servicio. Esta operación sólo modifica los recursos reservados convirtiéndolos en disponibles.

A continuación se muestra la signatura de la operación **cancelAllRemainingReservations**:

```
AvailableSysBandwidthSeqType cancelAllRemainingReservations(
              in ManagedEntityIdType nEManagedEntityId)
              raises (UnknownNE, CommFailure, AccessDenied);
```

El parámetro de entrada **nEManagedEntityId** se utiliza para identificar el elemento de red.

El valor devuelto es del tipo **AvailableSysBandwidthSeqType** (***tipo de secuencia de anchura de banda disponible en el sistema***) e indica la anchura de banda actualmente disponible para el elemento de red tras la supresión de la anchura de banda reservada.

9.11.1.7 getReservation (*obtener reserva*)

Esta operación se utiliza para averiguar de dónde procede una capacidad de reserva del NE.

A continuación se muestra la signatura de la operación **getReservation**:

```
ReservationInfoType getReservation (
               in ReservationIdType reservationId)
               raises (UnknownReservationId, AccessDenied);
```

El parámetro de entrada **reservationId** se utiliza para especificar la reserva.

El valor devuelto es del tipo **ReservationIdInfoType** (***tipo de información de Id de reserva***) e indica la información de conexión del servicio como parte de la petición de reserva con el **reservationId** proporcionado.

9.11.1.8 getAvailableSysBandwidth (*obtener anchura de banda disponible en el sistema*)

Esta operación se utiliza para determinar la capacidad remanente para reserva de conexiones de servicio o asignación a elementos de red.

A continuación se muestra la signatura de la operación **getAvailableSysBandwidth**:

```
AvailableSysBandwidthSeqType getAvailableSysBandwidth (
              in ManagedEntityIdType nEManagedEntityId)
              raises (UnknownNE, CommFailure, AccessDenied);
```

El parámetro de entrada **nEManagedEntityId** se utiliza para identificar el elemento de red.

El valor devuelto es del tipo **AvailableSysBandwidthSeqType** e indica la anchura de banda actualmente disponible para el elemento de red. Proporciona la anchura de banda disponible para cada puerto suministrado en el elemento de red caracterizado por la implementación del proveedor.

9.11.1.9 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema no tiene autorización de acceso al objeto de interfaz.

La excepción **CommFailure** (***fallo de comunicaciones***) se plantea cuando cae la RCD entre el sistema de gestión del proveedor y el OLT o se interrumpe la comunicación entre el OLT y el ONT u ONU.

La excepción **ConnectionCountExceeded** (***cómputo de conexiones sobrepasado***) se plantea cuando se excede el número máximo de conexiones para el puerto del OLT o de la PON con esta petición de prestación de servicio.

La excepción **InsufficientBW** (***anchura de banda insuficiente***) se plantea cuando la anchura de banda es insuficiente para el servicio solicitado.

La excepción **MaxSubtendingNodesExceeded** (***superado el número máximo de nodos subyacentes***) se plantea cuando se supera el máximo número de nodos subyacentes previsto para la interfaz PON identificada con esta petición de prestación del servicio.

La excepción **ProfileSuspended** (***perfil suspendido***) se plantea cuando el OMS o el operador suspenden el uso en el sistema de gestión del proveedor de los perfiles nombrados en la invocación.

La excepción **UnknownNE** (***NE desconocido***) se plantea cuando el OLT es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownPort** (***puerto desconocido***) se plantea cuando el puerto identificado es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownProfiles** (***perfiles desconocidos***) se plantea cuando el nombre del perfil proporcionado es desconocido para el sistema de gestión del proveedor y no puede recuperarse del repositorio de objetos de perfil.

La excepción **UnknownReservationId** (***Id de reserva desconocido***) se plantea cuando el sistema de gestión del proveedor no reconoce el ID de la reserva.

La excepción **UnknownServiceInstance** (***ejemplar de servicio desconocido***) se plantea cuando el ejemplar de servicio es desconocido para el sistema de gestión del proveedor.

9.12 Módulo SchedulerManagement (*gestión de planificador*)

Este servicio se utiliza para proporcionar interfaces OMS para los planificadores de gestión a fin de invocar diversas actividades. Una vez creado el planificador, el OMS puede lanzar peticiones de planificación de actividades tales como el envío de la MIB del NE, la transferencia en bloque, las pruebas o la descarga de soporte lógico, haciendo referencia al planificador. Los planificadores se definen en función de las necesidades del entorno de operaciones. Se supone que no hay planificadores nombrados ni establecidos automáticamente en la ejemplificación del sistema de gestión del proveedor. Las referencias a un planificador específico se efectuarán siempre mediante la etiqueta de usuario. Se utiliza una matriz de activación para describir la planificación. Los valores de la matriz se interpretan en base al valor de HourlyDailyWeeklyMonthlyInd.

9.12.1 Interfaz SchedulerMgr (*gestor del planificador*)

9.12.1.1 makeScheduler (*crear planificador*)

Esta operación crea un nuevo objeto planificador. A continuación, el OMS puede asociar actividades al objeto planificador haciendo referencia al nombre de la etiqueta de usuario del planificador.

A continuación se muestra la signatura de la operación **makeScheduler:**

```
void makeScheduler (
            in UserLabelType schedulerName,
            in GeneralizedTimeType startTime,
            in GeneralizedTimeType stopTime,
            in HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd,
            in TriggerTimeMatrixSeqType matrix)
            raises (InvalidStartTime, InvalidStopTime, DuplicateUserLabel,
            MatrixSchedulerTypeMismatch, AccessDenied, InvalidTrigger);
```

El parámetro de entrada **schedulerName** identifica el planificador al que puede hacerse referencia en distintas actividades. Los parámetros de entrada **startTime** y **stopTime** identifican el intervalo horario en el que se pueden aplicar las actividades planificadas. El parámetro de entrada **hourlyDailyWeeklyMonthlyInd** indica la frecuencia (horaria, diaria, semanal o mensual) con la que debe activarse la planificación. El parámetro de entrada **matrix** proporciona información de invocación específica de la planificación y depende del valor de parámetro **hourlyDailyWeeklyMonthlyInd**.

El cuadro 4 a continuación, proporciona un cuadro que muestra la dependencia de **hourlydailyWeeklyMonthlyInd** y de **matrix**:

Cuadro 4/Q.834.4 – Detalles de la matriz

Valor hourlyDailyWeeklyMonthlyInd	Valor de matrix
Horario	time (hora) – Cualquier valor entre 0 y 3600[7] dayOfWeek (día de la semana) – debe ser 'unspecified' (sin especificar) dayOfMonth (día del mes) – debe ser 0
Diario	time (hora) – Cualquier valor entre 0 y 86400[8] dayOfWeek – debe ser 'unspecified' dayOfMonth – debe ser 0
Semanal	Time (hora) – Cualquier valor entre 0 y 86400 dayOfWeek – debe ser distinto de 'unspecified' dayOfMonth – debe ser 0
Mensual	Time (hora) – Cualquier valor entre 0 y 86400 dayOfWeek – debe ser 'unspecified' dayOfMonth – debe se distinto de 0

El valor devuelto es del tipo **void**.

9.12.1.2 suspendScheduler (*suspender planificador*)

Esta operación se utiliza para suspender un planificador. Esencialmente establece el administrativeState de planificación pasando de 'unlocked' (*desbloqueado*) a 'locked' (*bloqueado*). Esto se aplicará a partir de la próxima iteración de planificación. Esta operación origina la suspensión de cualquier actividad planificada asociada.

A continuación se muestra la signatura de la operación **suspendScheduler**:

```
void suspendScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied);
```

El parámetro de entrada **schedulerName** se utiliza para identificar la planificación a suspender.

El valor devuelto es del tipo **void**.

9.12.1.3 resumeScheduler (*reanudar planificador*)

Esta operación se utiliza para reanudar un planificador suspendido. Esencialmente establece el administrativeState de planificación de 'locked' (*bloqueado*) a 'unlocked' (*desbloqueado*). Esto se aplica a partir de la siguiente iteración de planificación. Esta operación motiva la reanudación de cualquier actividad planificada asociada.

A continuación se muestra la signatura de la operación **resumeScheduler**:

```
void resumeScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied );
```

El parámetro de entrada **schedulerName** se utiliza para identificar la planificación a reanudar.

El valor devuelto es del tipo **void**.

---

[7] Representa el número de segundos desde la hora de comienzo del periodo de activación.

[8] Representa el número de segundos desde el día de comienzo del periodo de activación contando a partir de la medianoche anterior.

9.12.1.4 modifyTime (*modificar la hora*)

Esta operación se utiliza para modificar el **startTime** y el **stopTime** de una planificación. La utiliza el OMS para ampliar o acortar el periodo de vigencia de la planificación.

A continuación se muestra la signatura de la operación **modifyTime**:

```
void modifyTime (
            in UserLabelType schedulerName,
            in GeneralizedTimeType newStartTime,
            in GeneralizedTimeType newStopTime)
            raises (InvalidStartTime, InvalidStopTime, UnknownScheduler,
            AccessDenied);
```

El parámetro de entrada **schedulerName** se utiliza para identificar la planificación. El parámetro de entrada **newStartTime** proporciona la hora de comienzo del planificador mientras que el parámetro de entrada **newStopTime** proporciona la hora de detención del planificador. Si el valor de newStartTime es anterior a la hora actual del sistema, se provocará la excepción InvalidStartTime. El valor de 0 para **newStartTime** o para **newStopTime** indica que el valor anterior no debe modificarse.

El valor devuelto es del tipo **void**.

9.12.1.5 changeSchedulerName (*modificar el nombre del planificador*)

Esta operación se utiliza para modificar el nombre del objeto planificador. Si esta operación tiene éxito, el nuevo nombre del planificador entrará en vigor inmediatamente. Cualquier referencia a esta planificación tendrá que hacerse en base al nuevo nombre asignado.

Hay muchas actividades que pueden asociarse al mismo nombre de planificador. La modificación del nombre del planificador no repercutirá en las actividades asociadas al planificador. Todas las actividades asociadas al planificador modificado conservarán su asociación.

A continuación se muestra la signatura de la operación **changeSchedulerName**:

```
void changeSchedulerName (
            in UserLabelType oldSchedulerName,
            in UserLabelType newSchedulerName)
            raises (UnknownScheduler, DuplicateUserLabel, AccessDenied);
```

El parámetro de entrada **oldSchedulerName** se utiliza para identificar la planificación existente. El parámetro de entrada **newSchedulerName** se utiliza para identificar el nuevo nombre que se asignará a esta planificador.

El valor devuelto es del tipo **void**.

9.12.1.6 modifyTriggerTimes (*modificar horas de activación*)

Esta operación la utiliza el OMS para especificar nuevas horas de activación y de iteración para un planificador[9]. Si esta operación tiene éxito, cualquier actividad planificada asociada tendrá lugar en la nueva hora planificada a partir de la siguiente iteración de planificación.

A continuación se muestra la signatura de la operación **modifyTriggerTimes**:

```
void modifyTriggerTimes (
                  in UserLabelType schedulerName,
                  in HourlyDailyWeeklyMonthlyIndType
                  newHourlyDailyWeeklyMonthlyInd,
                  in TriggerTimeMatrixSeqType newMatrix)
```

[9] Sírvase consultar la operación makeScheduler en relación con los detalles de la dependencia de las variables utilizadas.

```
                raises (UnknownScheduler, MatrixSchedulerTypeMismatch,
                InvalidTrigger, AccessDenied );
```

El parámetro de entrada **schedulerName** se utiliza para identificar la planificación a modificar. El parámetro de entrada **newHourlyDailyWeeklyMonthlyInd** indica la nueva frecuencia con la que debe activarse la planificación. El parámetro de entrada **newMatrix** se utiliza para proporcionar la información de invocación específica de la nueva planificación. Sírvase consultar el cuadro 4 en relación con la dependencia de las variables. **newDailyWeeklyMonthlyInd** y **newMatrix** deben especificarse explícitamente.

El valor devuelto es del tipo **void**.

9.12.1.7 removeScheduler (*suprimir planificador*)

Esta operación se utiliza para suprimir un planificador. Esta operación sólo será admisible cuando no haya actividades asociadas a esta planificador.

A continuación se muestra la signatura de la operación **removeScheduler**:

```
void removeScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied, ScheduleInUse );
```

El parámetro de entrada **schedulerName** se utiliza para identificar la planificación a suprimir.

El valor devuelto es del tipo **void**.

9.12.1.8 retrieveScheduler (*recuperar planificador*)

Esta operación se utiliza para recuperar información sobre la planificación. El objeto de planificación se devolverá tras la invocación con éxito de esta operación.

A continuación se muestra la signatura de la operación **retrieveScheduler**:

```
SchedulerType retrieveScheduler (
              in UserLabelType schedulerName)
              raises (UnknownScheduler, AccessDenied);
```

El parámetro de entrada **schedulerName** se utiliza para identificar la planificación a visualizar.

El valor devuelto es del tipo **Scheduler** y proporciona la información siguiente: schedulerName (*nombre del planificador*); startTime (*hora de inicio*); stopTime (*hora de finalización*); hourlyDailyWeeklyMonthlyInd (*indicador horario diario semana mensual*); matrix (*matriz*); operationalState (*estado operacional*) y administrativeState (*estado administrativo*).

9.12.1.9 schedulerListGet (*obtener lista de planificadores*)

Esta operación se utiliza para recuperar los nombres de todos los planificadores existentes definidos en el sistema de gestión del proveedor.

A continuación se muestra la signatura de la operación **schedulerListGet**:

```
SchedulerSeqType schedulerListGet ()
              raises (AccessDenied);
```

No hay parámetros de entrada.

El valor devuelto es del tipo **SchedulerSeqType** y proporciona el listado deseado.

9.12.1.10 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema no tiene autorización de acceso al objeto de interfaz.

La excepción **DuplicateUserLabel** (***etiqueta de usuario duplicada***) se plantea cuando la etiqueta de usuario proporcionada en la petición se ha utilizado para etiquetar otro planificador. Dicho de otro modo, el sistema de gestión del proveedor se encarga de supervisar la asignación de etiquetas de usuario a los planificadores en su jurisdicción de gestión.

La excepción **InvalidStartTime** (***hora de comienzo no válida***) se plantea cuando la hora de comienzo especificada es incongruente con la matriz horaria de activación actual o con la hora de detención.

La excepción **InvalidStopTime** (***hora de detención no válida***) se plantea cuando la hora de detención especificada es incongruente con la matriz horaria de activación actual o con la hora de comienzo.

La excepción **InvalidTrigger** (***activador no válido***) se plantea cuando el activador especificado tiene valores que el planificador no puede interpretar.

La excepción **MatrixSchedulerTypeMismatch** (***desadaptación del tipo de planificador de matriz***) se plantea cuando la matriz horaria de activación está desadaptada con respecto al tipo de planificador nombrado.

La excepción **ScheduleInUse** (***planificación utilizándose***) la devolverá la operación removeScheduler cuando haya operaciones asociadas a la planificación.

La excepción **UnknownScheduler** (***planificador desconocido***) se plantea cuando no aparezca el planificador del nombre en cuestión.

9.13 Módulo ServiceProvisioning (*prestación del servicio*)

En este módulo el sistema de gestión del proveedor seleccionada puertos, facilidades y anchura de banda para poder completar los procesos de diseño, selección y asignación asociados a un servicio.

9.13.1 Interfaz ServiceProvisioner (*prestadora del servicio*)

9.13.1.1 provisionConnection (*suministrar conexión*)

Esta operación suministra una conexión entre dos puntos extremos cualesquiera y un sistema de acceso de fibra BPON. Puede establecerse una conexión entre una NNI y una UNI, entre dos UNI o entre dos NNI. En la figura 1 se representan estos puntos extremos.

A continuación se muestra la signatura de la operación **provisionConnection**:

```
ManagedEntityIdType provisionConnection(
                    in EndpointType endPointA,
                    in EndpointType endPointZ,
                    in NameSeqType networkCharacteristicsProfiles,
                    in ServiceInstanceIdType serviceInstanceId,
                    in AdministrativeStateType  administrativeState)
                    raises (UnknownNE, UnknownProfiles, UnknownPort,
                    InsufficientBW, ConnectionCountExceeded, CommFailure,
                    EquipmentFailure, ParameterViolation,AccessDenied,
                    InsufficientPONBW, ProfileSuspended,
                    ConnectionAlreadyExists);
```

Los parámetros de entrada **endPointA** y **endPointZ** identifican los puntos extremos de la petición de conexión. Un punto extremo se define por la estructura de datos que aparece en el cuadro 5.

Cuadro 5/Q.834.4 – Detalles del punto extremo

Nombre del campo	Definición	Sintaxis	Observaciones
portId (*Id de puerto*)	Especifica el puerto físico que contiene un punto extremo de la conexión	ManagedEntityIdType (*tipo de Id de entidad gestionada*)	Se supone que se trata de la entidad gestionada physicalPathTP para el puerto
endPointParameters (*parámetros del punto extremo*)	Identifica los parámetros específicos del ejemplar de servicio que contribuyen a reunir los componentes de conexión de la red	Cualquiera	Las estructuras deben suministrarse como parte de la implementación. Por ejemplo, una conexión ATM tendría los parámetros VPI y VCI
serviceCharacteristicsProfile (*perfil de características del servicio*)	Lista las referencias a los perfiles que caracterizan el servicio en el punto extremo	NameSeqType (*tipo de secuencia nombre*)	Como ejemplo se pueden citar AAL1Profile, AAL5Profile, ATMNetworkAccess Profile, UNIProfile, CESServiceProfile, DS1Profile, DS3Profile, EthernetProfile, AAL2Profile, LESProfile, SSCSParameterProfile1, SSCSParameterProfile2, VoiceServiceProfileAAL2, BridgedLANService Profile, MACBridgeServiceProfile (Nota)
NOTA – Las opciones específicas dependen de las características del servicio y de los equipos. Los perfiles se listan por el nombre proporcionado en el módulo Q834ProfileManager. El anexo D contiene ejemplos de relaciones.			

El parámetro de entrada **networkCharacteristicsProfiles** (*perfiles de características de red*) proporciona la lista de perfiles relacionados con el transporte que han de utilizarse en la prestación del servicio incluidos azTrafficDescriptorProfile y zaTrafficDescriptorProfile. Los perfiles de descriptor de tráfico az aparecen en el listado antes que los za. El parámetro **serviceInstanceId** contiene el identificador de ejemplar de servicio a utilizar como clave cuando se refiere a recursos de red asociados al servicio. El parámetro de entrada **administrativeState** especifica si la conexión de subred es capaz o no de cursar tráfico de abonado una vez ejecutada la operación.

El valor devuelto es del tipo **ManagedEntityIdType** y es un identificador de entidad gestionada que permite identificar la subnetworkConnection creada como resultado de esta petición.

9.13.1.2 provisionReservation (*suministrar reserva*)

Esta operación presta el servicio entre la NNI de un OLT y la UNI de un ONT o entre dos UNI en base a una reserva pendiente. Una vez suministrada la conexión, la anchura de banda reservada y el Id de reserva asociado se suprimen del sistema de gestión del proveedor debido a la asignación de los recursos reservados.

A continuación se muestra la signatura de la operación **provisionReservation**:

```
ManagedEntityIdType provisionReservation(
                    in ReservationIdType reservationId,
                    in AdministrativeStateType  administrativeState)
                    raises ( UnknownReservationId, AccessDenied);
```

El parámetro de entrada **reservationId** especifica el Id de la reserva. Apunta a toda la información (tal como el Id del ejemplar de servicio y los puntos extremos) a asociar a la conexión suministrada. El parámetro de entrada **administrativeState** especifica si la conexión de subred es capaz o no de cursar tráfico de abonado una vez ejecutada la operación.

El valor devuelto es del tipo **ManagedEntityIdType** y es un identificador de entidad gestionada de la subnetworkConnection creada como resultado de esta petición.

9.13.1.3 deleteConnection (*suprimir conexión*)

Esta operación cancela el servicio en las conexiones existentes.

A continuación se muestra la signatura de la operación **deleteConnection**:

```
void deleteConnection(
             in ManagedEntityIdType subnetworkConnectionId)
             raises (UnknownConnection, CommFailure, EquipmentFailure,
             AccessDenied);
```

El parámetro de entrada **subnetworkConnectionId** es la subred creada anteriormente mediante una petición de prestación de servicio.

El valor devuelto es del tipo **void**.

9.13.1.4 modifyConnection (*modificar conexión*)

Esta operación permite modificar un servicio existente.

A continuación se muestra la signatura de la operación **modifyConnection**:

```
ManagedEntityIdType modifyConnection (
                    in ManagedEntityIdType subnetworkConnectionId,
                    in ManagedEntityIdType  portB,
                    in NameSeqType newNetworkCharacteristicsProfiles,
                    in NameSeqType newServiceCharacteristicsProfiles)raises
                    (UnknownConnection, UnknownProfiles, InsufficientBW,
                    UnknownPort, AccessDenied, ProfileSuspended );
```

El parámetro de entrada **subnetworkConnectionId** es la subred creada anteriormente mediante la petición de prestación de servicio. El parámetro de entrada **portB** permite aclarar la direccionalidad de los perfiles de descriptores de tráfico nombrados en el siguiente parámetro de entrada, así como especificar dónde han de aplicarse los perfiles del servicio. Este puerto es un puerto A o un puerto Z de la petición de conexión original. El parámetro de entrada **newNetworkCharacteristicsProfiles** proporciona la lista modificada de los perfiles relacionados con la red que han de utilizarse para la prestación del servicio y en la que los perfiles de descriptores de tráfico "b-a-punto extremo opuesto" se enumeran antes que los perfiles de descriptor de tráfico "punto extremo opuesto-a-b". El parámetro de entrada **newServiceCharacteristicsProfiles** proporciona la nueva lista de perfiles relacionados con el servicio.

El valor devuelto es del tipo **ManagedEntityIdType** y es un identificador de entidad gestionada de la subnetworkConnection creada como resultado de esta petición.

9.13.1.5 suspendService (*suspender servicio*)

Esta operación desactiva el flujo de tráfico de usuario por la conexión de subred del servicio. La suspensión del servicio es totalmente equivalente al bloqueo del estado administrativo de la conexión de la subred.

A continuación se muestra la signatura de la operación **suspendService**:

```
void suspendService (
            in ServiceInstanceIdType serviceInstanceId,
            in GeneralizedTimeType startTime,
            in GeneralizedTimeType stopTime)
            raises (UnknownServiceInstance, AccessDenied, InvalidStartTime,
            InvalidStopTime);
```

El parámetro de entrada **serviceInstanceId** identifica la conexión de servicio a suspender. El parámetro de entrada **startTime** proporciona el instante en el que ha de suspenderse el servicio. El parámetro de entrada **stopTime** proporciona el instante en el que ha de reanudarse el servicio.

El valor devuelto es del tipo **void**.

9.13.1.6 resumeService (*reanudar servicio*)

Esta operación activa el flujo de tráfico de usuario por la conexión de subred de servicio. Esta operación puede invocarse ya sea tras la suspensión del servicio (véase la operación anterior) o cuando la conexión de servicio original se configura en el estado inactivo. El servicio ha de reanudarse inmediatamente.

A continuación se muestra la signatura de la operación **resumeService**:

```
void resumeService (
                in ServiceInstanceIdType serviceInstanceId)
                raises (UnknownServiceInstance, AccessDenied);
```

El parámetro de entrada **serviceInstanceId** identifica la conexión de servicio a reanudar.

El valor devuelto es del tipo **void**.

9.13.1.7 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando se deniega el acceso al NE por motivos de seguridad.

La excepción **CommFailure** (***fallo de comunicaciones***) se plantea cuando falla la comunicación entre el sistema de gestión del proveedor y los NE.

La excepción **ConnectionAlreadyExists** (***la conexión ya existe***) se plantea cuando ya existe una conexión de subred con los mismos puntos extremos.

La excepción **ConnectionCountExceeded** (***cómputo de conexión sobrepasado***) se plantea cuando el contador de conexión sobrepasa el cómputo de conexiones admisible.

La excepción **EquipmentFailure** (***fallo del equipo***) se plantea cuando la conexión solicitada no puede aplicarse a un recurso de red instalado por avería del NE.

La excepción **InsufficientBW** (***anchura de banda insuficiente***) se plantea cuando la anchura de banda es insuficiente para el servicio solicitado.

La excepción **InsufficientPONBW** (***anchura de banda de la PON insuficiente***) se plantea cuando la anchura de banda de PON disponible no es suficiente para soportar la prestación del servicio solicitado.

La excepción **InvalidStartTime** (***hora de comienzo no válida***) se plantea cuando la hora de comienzo especificada es incongruente con la hora actual o con la hora de detención.

La excepción **InvalidStopTime** (***hora de detención no válida***) se plantea cuando la hora de detención especificada es incongruente con la hora actual o con la hora de comienzo.

La excepción **ParameterViolation** (***violación de parámetro***) se plantea cuando los parámetros del punto extremo no concuerdan con las características de protocolo del puerto, o cuando los valores se encuentran fuera de rango o son duplicados no válidos.

La excepción **ProfileSuspended** (***perfil suspendido***) se plantea cuando el OMS o el operador suspenden el uso en el sistema de gestión del proveedor de los perfiles nombrados en la invocación.

La excepción **UnknownConnection** (***conexión desconocida***) se plantea cuando no se encuentra la conexión a suprimir.

La excepción **UnknownNE** (***NE desconocido***) se plantea cuando el sistema de gestión del proveedor no conoce el nombre del OLT.

La excepción **UnknownPort** (***puerto desconocido***) se plantea cuando el sistema de gestión del proveedor no conoce el ID del puerto de entrada.

La excepción **UnknownProfiles** (***perfiles desconocidos***) se plantea cuando el nombre del perfil proporcionado es desconocido para el sistema de gestión del proveedor y no puede recuperarse del repositorio de objetos de perfil.

La excepción **UnknownReservationId** (***Id de reserva desconocido***) se plantea cuando el identificador de reserva es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownServiceInstance** (***ejemplar de servicio desconocido***) se plantea cuando el identificador de servicio es desconocido para el sistema de gestión del proveedor.

9.14 Módulo Synchroniser (*sincronizador*)

Este módulo gestiona el proceso de sincronización entre el sistema de gestión del proveedor y un NE específico. El proceso de sincronización solicitado puede reducirse a un conjunto específico de listados de eventos actuales. Los detalles del método de detección de incongruencias y de los datos que se recuperan dependen de la implementación del proveedor. No obstante, este proceso origina la supresión de las incongruencias que puedan existir entre la información de gestión de la que dispone el NE y el modelo de información mantenido en el sistema de gestión del proveedor.

La operación de este módulo sólo puede invocarla un usuario privilegiado. El proceso de sincronización se efectúa sin garantías, en el sentido de que las incongruencias detectadas y corregidas antes de un plazo[10] definido por el sistema se conservan. El proceso de sincronización puede repercutir negativamente en la calidad de funcionamiento del sistema de gestión del proveedor antes de su compleción.

Cuando el elemento de red escogido es un OLT, el proceso de sincronización afecta a todo el sistema.

9.14.1 Interfaz NESynchroniser (*sincronizadora del NE*)

9.14.1.1 synchNE (*sincronizar NE*)

Esta operación inicia un proceso de sincronización entre el sistema de gestión del proveedor y un NE específico. Esta operación sólo se bloquea durante el tiempo que tarde el sistema de gestión del proveedor en validar el inicio del proceso y ejecuta la operación de sincronización en un segundo plano.

[10] Este plazo debe ser objeto de acuerdo entre el operador y el proveedor.

A continuación se muestra la signatura de la operación **synchNE**:

```
void synchNE(in ManagedEntityIdType nEManagedEntityId)
            raises (AccessDenied, CommFailure, UnknownNE, EquipmentFailure,
            BackupInProgress, SynchInProgress);
```

La entrada **nEManagedEntityId** identifica el NE a sincronizar con el sistema de gestión del proveedor.

El valor devuelto es del tipo **void**.

9.14.1.2 abortSynchNE (*abortar sincronización de NE*)

Esta operación aborta un proceso de sincronización en curso entre el sistema de gestión del proveedor y un NE específico. Las incongruencias entre el sistema de gestión del proveedor y el NE resueltas antes de la interrupción, se mantienen.

Esta operación sólo puede invocarla un usuario privilegiado.

A continuación se muestra la signatura de la operación **abortSynchNE**:

```
void   abortSynchNE(
             in ManagedEntityIdType nEManagedEntityId)
             raises (AccessDenied, CommFailure, UnknownNE, EquipmentFailure,
             NoSynchInProgress);
```

La entrada **nEManagedEntityId** identifica el NE cuya sincronización con el sistema de gestión del proveedor ha de abortarse.

El valor devuelto es del tipo **void**.

9.14.1.3 scheduleSynchNE (*planificar sincronización de NE*)

Esta operación planifica un proceso de sincronización entre el sistema de gestión del proveedor y un NE específico. El proceso de sincronización planificado lo activa el planificador de acuerdo con la definición previa del operador. Para un NE específico sólo puede asociarse a esta actividad un solo planificador como máximo.

A continuación se muestra la signatura de la operación **scheduleSynchNE**:

```
void scheduleSynchNE(
           in ManagedEntityIdType nEManagedEntityId,
           in UserLabelType schedulerName)
           raises (AccessDenied, UnknownNE, UnknownScheduler,
           InvalidScheduler);
```

La entrada **nEManagedEntityId** identifica el NE a sincronizar con el sistema de gestión del proveedor. La entrada **schedulerName** identifica el planificador a utilizar para invocar la operación de sincronización planificada del sistema de gestión del proveedor.

El valor devuelto es del tipo **void**.

9.14.1.4 modifyNESynchSchedule (*modificar planificación de sincronización de NE*)

Esta operación modifica la planificación de la sincronización del NE. Esta operación no interrumpe los procesos de sincronización en curso. Si tiene éxito, se aplica la nueva planificación en la siguiente iteración.

A continuación se muestra la signatura de la operación **modifyNESynch**:

```
void modifyNESynchSchedule(
            in ManagedEntityIdType nEManagedEntityId,
            in UserLabelType newSchedulerName)
            raises (AccessDenied, UnknownNE, UnknownScheduler,
            InvalidScheduler);
```

La entrada **nEManagedEntityId** identifica el NE en el que se activa la sincronización planificada. La entrada **newSchedulerName** identifica la planificación a aplicar.

El valor devuelto es del tipo **void**.

9.14.1.5 cancelScheduledSynchNE (*cancelar sincronización planificada de NE*)

Esta operación cancela todos los procesos de sincronización planificados para este NE. Esta operación no interrumpe los procesos de sincronización en curso.

A continuación se muestra la signatura de la operación **cancelScheduledSynchNE**:

```
void cancelScheduledSynchNE(
            in ManagedEntityIdType nEManagedEntityId )
            raises (AccessDenied, UnknownNE);
```

La entrada **nEManagedEntityId** identifica el NE en el que se activa la sincronización planificada.

El valor devuelto es del tipo **void**.

9.14.1.6 synchCurrentEventListings (*sincronizar listado actual de eventos*)

Esta operación inicia la sincronización entre el sistema de gestión del proveedor y un NE específico en cuanto a los elementos de un determinado listado de eventos actuales. El sistema de gestión del proveedor suele recuperar los valores actuales de los atributos de estado, situación y gestión, y efectúa un seguimiento de los mismos a través del listado resumen de eventos actuales del sistema. Si algún proceso automático de recuperación del sistema muestra que el listado no está actualizado con las condiciones actuales del sistema, se modifica el listado (mediante la supresión de entradas o la inserción de otras nuevas) a fin de corregirlo. Esta operación es una versión manual en respuesta a una petición del OMS. Durante esta operación, no puede invocarse la excepción SynchInProgress para ninguna otra operación.

A continuación se muestra la signatura de la operación **synchCurrentEventListings**:

```
void synchCurrentEventListings(
                  in ManagedEntityIdType nEManagedEntityId,
                  in CurrentListingSeqType currentListingTypeList)
                  raises (AccessDenied, CommFailure, DCNTimeout, UnknownNE,
                  EquipmentFailure, Timeout);
```

La entrada **nEManagedEntityId** identifica el NE de sincronización. La entrada **currentListingTypeList** identifica la lista de eventos actuales que han de sincronizarse entre el sistema de gestión del proveedor y el NE específico. El sistema de gestión del proveedor recupera los valores indicados del listado de eventos actuales del NE.

El valor devuelto es del tipo **void**.

9.14.1.7 scheduledSynchNEListGet (*obtener listado de sincronización planificada del NE*)

Esta operación se utiliza para recuperar los nombres de todos los NE con sincronización planificada.

A continuación se muestra la signatura de la operación **scheduledSynchNEListGet**:

```
ScheduledSynchNESeqType scheduledSynchNEListGet ()
                  raises (AccessDenied);
```

No hay parámetros de entrada.

El valor devuelto es del tipo **ScheduledSynchNESeqType** (***tipo de secuencia de sincronización planificada del NE***) y proporciona el listado de los NE deseados.

9.14.1.8 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema no tiene autorización de acceso al objeto de interfaz.

La excepción **BackupInProgress** (***copia de seguridad en curso***) se plantea cuando la petición de sincronización se emite mientras se está ejecutando una copia de seguridad.

La excepción **CommFailure** (***fallo de comunicaciones***) se plantea cuando cae la RCD entre el sistema de gestión del proveedor y el OLT o se interrumpe la comunicación entre el OLT y el ONT u ONU subyacente.

La excepción **DCNTimeout** (***fin de temporización de la RCD***) se plantea cuando el enlace de comunicaciones de la RCD entre uno de los NE como mínimo y el sistema de gestión del proveedor está tan congestionado que no puede transferirse la información del estado o situación actual dentro de un plazo definido por el sistema.

La excepción **EquipmentFailure** (***fallo del equipo***) se plantea cuando el equipo del que se está realizando la copia de seguridad está averiado.

La excepción **InvalidScheduler** (***planificador no válido***) se plantea cuando el planificador en cuestión es inadecuado para ser utilizado en esta operación o bien es obsoleto.

La excepción **SynchInProgress** (***sincronización en curso***) se plantea cuando se solicita un nuevo synchNe mientras se ejecuta otro synchNe o scheduledSynchNe.

La excepción **SynchNotScheduled** (***sincronización no planificada***) se plantea si no hay ninguna sincronización planificada para el NE en un momento determinado.

La excepción **UnknownNE** (***NE desconocido***) se plantea cuando el NE mencionado en la petición es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownScheduler** (***planificador desconocido***) se plantea cuando no se encuentra el nombre del planificador en cuestión.

La excepción **Timeout** (***fin de temporización***) se plantea cuando el proceso especificado ha superado el plazo por defecto definido por el sistema.

La excepción **NoSynchInProgress** (***no hay sincronización en curso***) se plantea cuando no se está ejecutando ningún proceso de sincronización.

9.15 Módulo Test (*pruebas*)

Este servicio se utiliza para proporcionar interfaces al operador u OMS a fin de ejecutar procedimientos de prueba dirigidos o planificados. Mediante este servicio se especifican pruebas tales como las de bucle de células ATM OAM, el establecimiento de bucles de interfaz en tarjetas de abonado o en tarjetas de interfaz de red OLT y las pruebas de continuidad ATM. Las pruebas pueden ser planificadas o invocadas manualmente a raíz de una avería identificada o de una queja del servicio por parte de abonado. La interfaz TestActionPerformer proporciona las operaciones para ejecutar las pruebas aplicables a los recursos de red.

9.15.1 Interfaz TestActionPerformer (*ejecutora de acción de prueba*)

9.15.1.1 aTMLoopback (*bucle ATM*)

Esta operación se utiliza para invocar una prueba de bucle ATM OAM. La prueba de bucle ATM es unidireccional.

A continuación se muestra la signatura de la operación **aTMLoopback**:

```
AggregateATMLoopbackResultSeqType aTMLoopback (
                        in UserIdType testRequestorId,
                        in ManagedEntityIdType ctp,
                        in ATMLoopbackInfoType aTMLoopbackInfo,
                        in TestIterationNumType testIterationNum,
                        in ServiceInstanceIdType serviceInstanceId)
                        raises (AccessDenied, CommFailure, UnknownManagedEntity,
                        NotAvailableForTest, InvalidLocationId,
                        InvalidDirection);
```

El parámetro de entrada **testRequestorId** (***Id del solicitante de la prueba***) se utiliza para identificar el iniciador de la prueba de bucle ATM. El parámetro de entrada **ctp** se utiliza para identificar unívocamente el CTP del punto de inyección de células. El parámetro de entrada **aTMLoopbackInfo** (***información de bucle ATM***) se utiliza para proporcionar información específica de la prueba ATM. El LoopbackLocationId (*Id de situación del bucle*) forma parte del aTMLoopbackInfo (*información del bucle ATM*). El parámetro de prueba **testIterationNum** (***número de iteración de la prueba***) identifica el número de iteración de la prueba de bucle ATM. El parámetro de prueba **serviceInstanceId** (***Id de ejemplar de servicio***) identifica el posible ejemplar de servicio asociado a la petición de prueba de bucle.

Cuando se especifica aTMLoopbackInfo, los únicos valores aplicables a la direccionalidad son 'egress' (*salida*) o 'ingress' (*entrada*). La direccionalidad se especifica como parte de la **aTMLoopbackInfo**.

El LoopbackLocationId identifica los puntos a lo largo de la conexión virtual en los que se encuentra el bucle. El transmisor utiliza el valor por defecto de todo unos para indicar el punto extremo. Cuando segmentCellInd tiene el valor 'false', el LoopbackLocationId debe ponerse al valor por defecto. Si no se proporciona LoopbackLocationId se supone que hay un punto extremo del segmento designado para el flujo asociado al ctp de inyección.

Para el ctp, todos ceros indica una petición de bucle dirigida a todos los puntos de conexión con un LoopbackLocationId. Todo unos indica una petición de bucle dirigida al punto extremo (punto extremo del segmento o conexión). 'x6A'H indica que no hay CP designado para el bucle y que, por consiguiente, no debe ejecutarse ningún bucle. Los demás valores de LoopbackLocationId indican peticiones de bucle dirigidas a posiciones específicas de LoopbackLocationId.

El valor devuelto es del tipo **AggregateATMLoopbackResultSeqType** (***tipo de secuencia resultado de bucle ATM agregado***) y proporciona información sobre la prueba, concretamente el loopbackingLLID, responseTime en microsegundos y el éxito o fracaso de cada iteración.

9.15.1.2 initializeContinuityCheck (*inicializar verificación de continuidad*)

Esta operación se utiliza para preparar la verificación de continuidad en ATM. La creación de un entorno de prueba de continuidad no inicia forzosamente dicha prueba sino que planifica su iniciación. Cuando se crea con éxito una verificación de continuidad, el sistema devuelve un identificador único para la prueba.

Esta operación sólo se refiere al montaje de la prueba. Una vez ejecutada la prueba, se disparará una alarma en caso de fallo.

A continuación se muestra la signatura de la operación **initializeContinuityCheck**:

```
CCSetUpIdType initializeContinuityCheck(
                  in UserIdType testRequestorId,
                  in ManagedEntityIdType sourceCtp,
                  in ATMContinuityCheckInfoType aTMContinuityCheckInfo,
                  GeneralizedTimeType   stopTime,
                  in ServiceInstanceIdType serviceInstanceId)
```

```
                    raises (AccessDenied, CommFailure, UnknownManagedEntity,
                    NotAvailableForTest, InvalidStartTime, InvalidStopTime,
                    InvalidDirection);
```

El parámetro de entrada **testRequestorId** se utiliza para identificar el iniciador de la prueba. Los parámetros de entrada **sourceCtp** y **sinkCtp** se utilizan para identificar el punto A de la prueba de continuidad. El parámetro de entrada **aTMContinuityCheckInfo** se utiliza para proporcionar información específica de la prueba de continuidad. La direccionalidad debe tener el valor 'BothDirections' (*ambos sentidos*) y segmentCellInd debe ser 'true' (*verdadero*) para la prueba de continuidad de segmentos y 'false' (*falso*) para la de extremo a extremo. El parámetro de entrada **stopTime** (***hora de detención***) proporciona información sobre la duración de la prueba que debe ejecutarse. El parámetro de entrada **serviceInstanceId** identifica los posibles ejemplares de servicio asociados a la petición de prueba de bucle.

El valor devuelto es del tipo **CCSetUpIdType** e identifica unívocamente la prueba establecida. El CCSetUpId para la prueba montada se conserva hasta alcanzar stopTime o hasta su cancelación explícita mediante la operación terminateContinuityCheck.

9.15.1.3 terminateContinuityCheck (*cancelar verificación de continuidad*)

Esta operación se utiliza para suprimir el entorno de prueba de continuidad en ATM. Si la prueba de continuidad ya se hubiera iniciado (es decir se hubiera alcanzado startTime), la prueba de continuidad detendría su ejecución y se suprimiría.

A continuación se muestra la signatura de la operación **terminateContinuityCheck**:

```
void terminateContinuityCheck(
                    in CCSetUpIdType cCSetUpId)
                    raises (AccessDenied, CommFailure, UnknownTest);
```

El parámetro de entrada **CCSetUpId** se utiliza para identificar la prueba a cancelar.

El valor devuelto es del tipo **void**.

9.15.1.4 scheduleResourceSelfTest (*planificar autocomprobación de recursos*)

Esta operación se utiliza para planificar la autocomprobación de recursos. Esta operación la utiliza el OMS para montar autocomprobaciones de los recursos que deban ejecutarse periódicamente. La existencia de un montaje de objeto planificador es uno de los requisitos previos a la iniciación de esta operación.

A continuación se muestra la signatura de la operación **scheduleResourceSelfTest**:

```
TestTrackingObjectIdType scheduleResourceSelfTest(
                in UserIdType testRequestorId,
                in ManagedEntityIdType targetNE,
                in unsigned long timeOutPeriod, //In seconds.
                in ResourceSelfTestInfoSeqType specificTestInfo,
                in UserLabelType schedulerName)
                raises (AccessDenied, UnknownNE, UnknownScheduler,
                InvalidScheduler, InvalidTimeoutPeriod, InvalidTestOperations);
```

El parámetro de entrada **testRequestorId** se utiliza para identificar el iniciador de la prueba. El parámetro de entrada **targetNE** identifica el elemento de red que ejecutará la autocomprobación. El parámetro de entrada **timeOutPeriod** identifica el máximo plazo de ejecución de la prueba sobre el recurso, permitido por el sistema. El parámetro de entrada **specificTestInfo** se utiliza para proporcionar información de autocomprobación específica del proveedor, relativa a cada tipo de prueba de diagnóstico a ejecutar; esta información se facilita al operador en la documentación del sistema. El parámetro de entrada **schedulerName** se utiliza para hacer referencia al planificador aplicable a esta prueba.

El valor devuelto es del tipo **TestTrackingObjectIdType** (***tipo de Id de objeto de seguimiento de prueba***) e identifica unívocamente la prueba planificada. El objeto de seguimiento de la prueba existe hasta su cancelación explícita por medio de la operación terminateScheduledResourceSelfTest o si se alcanza la hora final del planificador.

Los resultados de la autocomprobación se anotan en el registro histórico. El tipo de datos **ResourceSelfTestResultSeqType** (***tipo de secuencia resultado autocomprobación recursos***) define parte de la información que se anota en el registro histórico.

9.15.1.5 modifyResourceSelfTestSchedule (*modificar planificación de autocomprobación de recursos*)

Esta operación se utiliza para modificar la planificación de una autocomprobación de recursos ejecutada periódicamente. De tener éxito, se modifica la iniciación de la autocomprobación de recursos en la siguiente iteración.

A continuación se muestra la signatura de la operación **modifyResourceSelfTestSchedule**:

```
void   modifyResourceSelfTestSchedule (
             in TestTrackingObjectIdType testTrackingObjectId,
             in UserLabelType newSchedulerName)
             raises (UnknownTest, UnknownScheduler, InvalidScheduler,
             AccessDenied);
```

El parámetro de entrada **testTrackingObjectId** (***Id de objeto de seguimiento de prueba***) se utiliza para identificar la prueba planificada. El parámetro de entrada **testTrackingObjectId** se utiliza para identificar la nueva planificación.

El valor devuelto es del tipo **void**.

9.15.1.6 cancelScheduledResourceSelfTest (*cancelar autocomprobación de recursos planificada*)

Esta operación se utiliza para cancelar una autocomprobación de recursos planificada periódicamente. De tener éxito, esta operación cancela la prueba antes de su iniciación en la siguiente hora de activación.

A continuación se muestra la signatura de la operación **cancelScheduledResourceSelfTest**:

```
void  cancelScheduledResourceSelfTest (
            in TestTrackingObjectIdType testTrackingObjectId)
            raises (UnknownTest, UncontrolledTestInProgress, AccessDenied);
```

El parámetro de entrada **testTrackingObjectId** se utiliza para identifica la prueba planificada que ha de cancelarse.

El valor devuelto es del tipo **void**.

9.15.1.7 conductResourceSelfTest (*ejecutar autocomprobación de recursos*)

Esta operación se utiliza para iniciar la autocomprobación de recursos como consecuencia de la identificación de una avería del sistema o de una queja de servicio por parte del abonado. Los resultados de la prueba se anotan en el registro histórico del sistema de gestión del proveedor.

A continuación se muestra la signatura de la operación **conductResourceSelfTest**:

```
TestTrackingObjectIdType conductResourceSelfTest (
              in UserIdType testRequestorId,
              in ManagedEntityIdType targetNE,
              in unsigned long timeOutPeriod, //In seconds.
              in ResourceSelfTestInfoSeqType specificTestInfo)
              raises (AccessDenied, CommFailure, UnknownNE,
              InvalidTimeoutPeriod, InvalidTestOperations);
```

El parámetro de entrada **testRequestorId** se utiliza para identificar el iniciador de la prueba. El parámetro de entrada **targetNE** identifica el elemento de red que ha de ejecutar la autocomprobación. El parámetro de entrada **timeOutPeriod** (***periodo de temporización***) identifica el plazo durante el que el sistema intentará iniciar la prueba. El parámetro de entrada **specificTestInfo** (***información específica de la prueba***) se utiliza para facilitar información específica de la autocomprobación con detalles sobre cada autocomprobación de diagnóstico a realizar.

El valor devuelto es del tipo **TestTrackingObjectId** (***Id de objeto de seguimiento de prueba***) y proporciona un mecanismo que permite al operador terminar una autocomprobación de recursos controlada. Si la autocomprobación de recursos no es controlada el sistema de gestión del proveedor devuelve el valor 0.

9.15.1.8 terminateResourceSelfTest (*terminar autocomprobación de recursos*)

Esta operación termina una autocomprobación de recursos en ejecución.

A continuación se muestra la signatura de la operación **terminateResourceSelfTest**:

```
ResourceSelfTestResultSeqType  terminateResourceSelfTest (
            in TestTrackingObjectIdType testTrackingObjectId)
            raises (UnknownTest, UncontrolledTestInProgress, AccessDenied);
```

El parámetro de entrada **testTrackingObjectId** se utiliza para identificar la prueba que ha de terminarse.

El valor devuelto es del tipo **ResourceSelfTestResultSeqType** (***tipo de secuencia de resultado de autocomprobación de recursos***) y proporciona los resultados intermedios de la prueba que puedan estar disponibles.

9.15.1.9 initiateLoopback (*iniciar bucle*)

Esta operación se utiliza para iniciar un bucle en un servicio. Por ejemplo, puede ejecutarse un NearEndLineLoopback (*bucle próximo al extremo de la línea*) DS1 [en el que caso de poder soportar un extremo remoto] o un FacilityLoop (*bucle de facilidad*) de SONET.

A continuación se muestra la signatura de la operación **initiateLoopback**:

```
LoopbackTrackingObjectIdType    initiateLoopback (
                                in UserIdType testRequestorId,
                                in ManagedEntityIdType loopingCtp,
                                in long duration, //In minutes.
                                in DirectionalityType directionality,
                                in LoopbackTestType loopbackTest
                                in ServiceInstanceIdType serviceInstanceId)
                                raises (AccessDenied, CommFailure,
                                UnknownManagedEntity, NotAvailableForTest);
```

El parámetro de entrada **testRequestorId** se utiliza para identificar el iniciador de la prueba. El parámetro de entrada **loopingCtp** se utiliza para identificar el CTP sobre el que ha de ejecutarse el bucle. El parámetro de entrada **duration** define el tiempo en segundos durante el que el bucle debe estar activo. El parámetro de entrada **directionality** indica si el bucle debe realizarse para tráfico de entrada, de salida o en ambos sentidos. El parámetro de entrada **loopbackTest** se utiliza para identificar el tipo de prueba de bucle específico. El parámetro de entrada **serviceInstanceId** se utiliza para especificar el servicio asociado a esta operación de bucle.

El valor devuelto es del tipo **LoopbackTrackingObjectIdType** (***tipo de Id de objeto de seguimiento de bucle***) e identifica unívocamente la prueba de bucle iniciada. Una vez completado

el ciclo de ejecución del bucle (es decir una vez transcurrida la prueba), el objeto ya no estará disponible.

9.15.1.10 terminateLoopback (*terminar bucle*)

Esta operación se utiliza para cancelar un bucle en ejecución.

A continuación se muestra la signatura de la operación **terminateLoopback**:

```
void terminateLoopback  (
                        in LoopbackTrackingObjectId loopbackTrackingObjectId)
                        raises (UnknownTest, AccessDenied);
```

El parámetro de entrada **loopbackTrackingObjectId** (***Id de objeto de seguimiento de bucle***) se utiliza para identificar el bucle a terminar.

El valor devuelto es del tipo **void**.

9.15.1.11 getLoopbackInfo (*obtener información de bucle*)

Esta operación se utiliza para recuperar la información del bucle correspondiente a un punto concreto de terminación de la conexión.

A continuación se muestra la signatura de la operación **getLoopbackInfo**:

```
LoopbackInfoType getLoopbackInfo (
                        in ManagedEntityIdType cTP)
                        raises (UnknownManagedEntity, AccessDenied);
```

El parámetro de entrada **cTP** se utiliza para identificar la posible posición del bucle.

El valor devuelto es del tipo **LoopbackInfoType** y especifica el tipo de bucle y su direccionalidad.

9.15.1.12 getLoopbackInfoByNE (*obtener información de bucle por NE*)

Esta operación se utiliza para recuperar la posición y los detalles de cada punto de conexión de un NE que se encuentra en modo de bucle.

A continuación se muestra la signatura de la operación **getLoopbackInfoByNE**:

```
LoopbackInfoSeqType getLoopbackInfoByNE    (
                        in ManagedEntityIdType nEId)
                        raises (UnknownManagedEntity, AccessDenied);
```

El parámetro de entrada **nEId** se utiliza para identificar el recurso de red.

El valor devuelto es del tipo **LoopbackInfoSeqType** y proporciona un listado de posiciones del bucle, tipos de bucle y su direccionalidad.

9.15.1.13 getTestStatus (*obtener estado de la prueba*)

Esta operación se utiliza para recuperar el estado de un bucle en ejecución.

A continuación se muestra la signatura de la operación **getTestStatus**:

```
StatusValueType getTestStatus (
                    in LoopbackTrackingObjectIdType   id)
                    raises (AccessDenied, UnknownTest);
```

El parámetro de entrada **id** especifica el montaje de la prueba de interés.

El valor devuelto es del tipo **StatusValueType** e indica el estado de la acción de bucle.

9.15.1.14 scheduledTestNEListGet (*obtener lista de pruebas de NE planificadas*)

Esta operación se utiliza para recuperar la lista de todas las pruebas de NE planificadas.

A continuación se muestra la signatura de la operación **scheduledTestNEListGet**:

```
ScheduledTestNESeqType scheduledTestNEListGet ()
                    raises (AccessDenied);
```

No hay parámetros de entrada.

El valor devuelto es del tipo **ScheduledTestNESeqType** y proporciona el listado deseado.

9.15.1.15 testHistoryByManagedEntity (*comprobar historia por entidad gestionada*)

Esta operación se utiliza para recuperar la historia de la prueba asociada a una entidad gestionada. Esta historia debe conservarse mientras lo necesite el operador.

A continuación se muestra la signatura de la operación **testHistoryByManagedEntity**:

```
TestHistorySeqType testHistoryByManagedEntity (
                    in ManagedEntityIdType managedEntityId)
                    raises (AccessDenied, UnknownManagedEntity);
```

El parámetro de entrada **managedEntityId** especifica la entidad gestionada de referencia.

El valor devuelto es del tipo **TestHistorySeqType** y proporciona el listado deseado.

9.15.1.16 testHistoryByServiceInstance (*comprobar historia por ejemplar de servicio*)

Esta operación se utiliza para recuperar la historia de la prueba asociada a un ejemplar de servicio. Esta historia debe conservarse mientras lo necesite el operador.

A continuación se muestra la signatura de la operación **testHistoryByServiceInstance**:

```
TestHistorySeqType testHistoryByServiceInstance (
                    in ServiceInstanceIdType serviceInstanceId)
                    raises (AccessDenied, UnknownServiceInstance);
```

El parámetro de entrada **serviceInstanceId** especifica el ejemplar de servicio de referencia.

El valor devuelto es del tipo **TestHistorySeqType** (*tipo de secuencia de historia de prueba*) y proporciona el listado deseado.

9.15.1.17 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el sistema no tiene autorización de acceso al objeto de interfaz.

La excepción **CommFailure** (***fallo de comunicaciones***) se plantea cuando cae la RCD entre el sistema de gestión del proveedor y el OLT o se interrumpe la comunicación entre el OLT y el ONT de origen.

La excepción **InvalidDirection** (***dirección no válida***) se plantea cuando la direccionalidad especificada no es válida para la prueba en cuestión.

La excepción **InvalidLocationId** (***Id de posición no válido***) se plantea cuando el LLID especificado no es válido.

La excepción **InvalidScheduler** (***planificador no válido***) se plantea cuando el planificador en cuestión es inadecuado para ser utilizado en esta operación o bien es obsoleto.

La excepción **InvalidStartTime** (***hora de comienzo no válida***) se plantea cuando la hora de comienzo especificada es incoherente con la matriz horaria de activación actual o con la hora de detención.

La excepción **InvalidStopTime** (***hora de detención no válida***) se plantea cuando la hora de detención especificada es incoherente con la matriz horaria de activación actual o con la hora de comienzo.

La excepción **InvalidTestOperations** (***operaciones de prueba no válidas***) se plantea cuando la operación de autocomprobación solicitada no es válida.

La excepción **InvalidTimeOutPeriod** (***periodo de temporización no válido***) se plantea cuando el periodo de temporización designado viola la definición de valores válidos.

La excepción **NotAvailableForTest** (***no disponible para la prueba***) se plantea cuando el sourceCTP (*CTP fuente*) no puede llegar a montar una prueba de verificación de continuidad con el sinkCTP (*CTP colector*).

La excepción **UncontrolledTestInProgress** (***prueba incontrolada en curso***) se plantea cuando no puede cancelarse la autocomprobación debido a que hay una prueba incontrolada.

La excepción **UnknownNE** (***NE desconocido***) se plantea cuando el NE mencionado en la petición es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownManagedEntity** (***entidad gestionada desconocida***) se plantea cuando la entidad gestionada especificada es desconocida para el sistema de gestión del proveedor.

La excepción **UnknownServiceInstance** (***ejemplar de servicio desconocido***) se plantea cuando el ejemplar de servicio especificado es desconocido para el sistema de gestión del proveedor.

La excepción **UnknownScheduler** (***planificador desconocido***) se plantea cuando el nombre del planificador en cuestión no aparece.

La excepción **UnknownTest** (***prueba desconocida***) se plantea cuando la prueba especificada por el trackingId no es conocida en el sistema de gestión del proveedor.

9.16 Módulo FileTransfer (*transferencia de ficheros*)

Este módulo se encarga de la gestión de la transferencia en diferido de los registros almacenados en cualquier archivo de corto plazo en el sistema de gestión del proveedor. Soporta el seguimiento y supervisión subsiguientes del proceso de transferencia de ficheros mediante la utilización del TransferTrackingObjectId (*Id de objeto de seguimiento de transferencia*). El sistema de gestión del proveedor anota en el registro histórico los resultados satisfactorios o insatisfactorios de la transferencia de ficheros. Cualquier petición de transferencia de ficheros viene acompañada de la correspondiente acreditación de seguridad mediante la que se permite al sistema de gestión del proveedor comunicarse con el servidor de destino. La transferencia de ficheros puede planificarse con anticipación. Los objetos de seguimiento de las transferencias los suprime automáticamente el sistema de gestión del proveedor una vez terminada la transferencia de ficheros asociada y los resultados de la misma (satisfactorios o insatisfactorios) se anotan en el registro histórico de compleción.

9.16.1 Interfaz TransferMgr (*gestor de transferencias*)

9.16.1.1 fileTransfer (*transferencia de fichero*)

Esta operación inicia inmediatamente la transferencia de un fichero.

A continuación se muestra la signatura de la operación **fileTransfer**:

```
TransferTrackingObjectIdType fileTransfer(
                        in ManagedEntityIdType recordSetId,
                        in DCNAddressType destinationServerAddr,
                        in UserIdType userId,
                        in PasswordType password,
                        in FilenameType destinationFile,
                        in boolean overwriteExistingFile)
                        raises (AccessDenied, CommFailure, UnknownRecordSet,
                        UnknownDestinationServer );
```

El parámetro de entrada **recordSetId** identifica el archivo de corto plazo del que han de extraerse los datos para la transferencia del fichero. El parámetro de entrada **destinationServerAddr** (***dirección del servidor de destino***) identifica la dirección de la red de comunicación de datos correspondiente al servidor de destino de la transferencia del fichero. Los parámetros de entrada **userId** y **password** proporcionan el mecanismo de clave de acceso al servidor de destino (suponiendo necesaria esta acreditación de seguridad). El parámetro de entrada **destinationFile** proporciona la información completa de la ruta de acceso del fichero transferido. Finalmente, el parámetro **overwriteExistingFile** (***sustituir fichero existente***) indica si se permite o no al fichero de la transferencia sustituir un fichero preexistente que tenga la misma ruta de acceso.

El valor devuelto es del tipo **TransferTrackingObjectIdType** (***tipo de Id objeto de seguimiento de transferencia***) y proporciona una clave de correlación a utilizar cuando se intenta observar el estado de la transferencia diferida de datos desde el archivo de corto plazo en algún instante posterior.

9.16.1.2 scheduleFileTransfer (*planificar transferencia de ficheros*)

Esta operación permite planificar una transferencia de ficheros futura. Todo el contenido del archivo de corto plazo especificado se extrae de la transferencia del fichero. No se establecen hipótesis sobre la eliminación del archivo dependiendo de si la transferencia de los datos del fichero se ha realizado con éxito o no. Por contra, el archivo se borra en base a los acuerdos entre proveedor y operador relativos a la política de retención de la información archivada.

A continuación se muestra la signatura de la operación **scheduleFileTransfer**:

```
TransferTrackingObjectIdType scheduleFileTransfer (
                        in ManagedEntityIdType recordSetId,
                        in DCNAddressType destinationServerAddr,
                        in UserIdType userId,
                        in PasswordType password,
                        in FilenameType destinationFile,
                        in boolean overwriteExistingFile,
                        in UserLabelType schedulerName)
                        raises (AccessDenied,UnknownRecordSet,
                        UnknownDestinationServer, UnknownScheduler,
                        InvalidScheduler);
```

El parámetro de entrada **recordSetId** (***Id del conjunto de registros***) identifica el archivo a corto plazo del que han de extraerse los datos para la transferencia del fichero. El parámetro de entrada **destinationServerAddr** (***dirección del servidor de destino***) identifica la dirección en la red de comunicación de datos del servidor de destino de la transferencia del fichero. Los parámetros de entrada **userId** y **password** proporcionan el mecanismo de acceso al servidor de destino (suponiendo necesaria dicha acreditación de seguridad). El parámetro de entrada **destinationFile** proporciona la ruta de acceso completa del fichero transferido. El parámetro **overwriteExistingFile** (***sustituir fichero existente***) indica si se debe permitir o no a la transferencia de ficheros sustituir un fichero preexistente que tenga la misma ruta de acceso de. Por último, el parámetro de entrada **schedulerName** es la información de planificación asociada a la transferencia de ficheros realizada, en base a la cual se efectúa la misma.

El valor devuelto es del tipo **TransferTrackingObjectIdType** (***tipo de Id de objeto de seguimiento de transferencia***) y proporciona una clave de correlación a utilizar cuando se intenta observar el estado de la transferencia de datos en diferido desde el archivo a corto plazo, en algún instante posterior.

9.16.1.3 modifyFileTransferSchedule (*modificar planificación de transferencia de ficheros*)

Esta operación modifica la planificación de la transferencia de ficheros. Si tiene éxito, se aplica una nueva planificación en la siguiente iteración.

A continuación se muestra la signatura de la operación **modifyFileTransferSchedule**:

```
void modifyFileTransferSchedule (
                        in TransferTrackingObjectIdType
                        transferTrackingObjectId,
                        in UserLabelType newSchedulerName)
                        raises (AccessDenied, UnknownTransferProcess,
                        UnknownScheduler, InvalidScheduler);
```

El parámetro de entrada **transferTrackingObjectId** (***Id de objeto de seguimiento de transferencia***) identifica la actividad de transferencia de ficheros planificada. El parámetro de entrada **newSchedulerName** es la nueva información de planificación a asociar a la transferencia de ficheros.

El valor devuelto es del tipo **void**.

9.16.1.4 cancelScheduledFileTransfer (*cancelar transferencia de ficheros planificada*)

Esta operación permite cancelar la planificación de la transferencia de ficheros. De tener éxito, la actividad se cancela en la siguiente iteración.

A continuación se muestra la signatura de la operación **cancelScheduledFileTransfer**:

```
void cancelScheduledFileTransfer (
                  in TransferTrackingObjectIdType transferTrackingObjectId)
                  raises (AccessDenied, UnknownTransferProcess);
```

El parámetro de entrada **transferTrackingObjectId** (***ID de objeto de seguimiento de transferencia***) identifica la actividad de transferencia de ficheros planificada.

El valor devuelto es del tipo **void**.

9.16.1.5 getStatus (*obtener estado*)

Esta operación permite que el cliente verifique el estado de la transferencia antes de su compleción por medio de una clave.

A continuación se muestra la signatura de la operación **getStatus**:

```
StatusValueType getStatus (
                  in TransferTrackingObjectIdType transferTrackingObjectId)
                  raises (UnknownTransferProcess, AccessDenied);
```

El parámetro de entrada **transferTrackingObjectId** (***Id de objeto de seguimiento de transferencia***) identifica la clave de un proceso concreto de transferencia de ficheros. El operador especifica esta información para obtener la información de estado de dicho proceso de transferencia de ficheros.

El valor devuelto es del tipo **StatusValueType** y proporciona el estado del proceso de transferencia de ficheros.

9.16.1.6 fileTransferHistoryListGet (*obtener lista histórica de transferencia de ficheros*)

Esta operación se utiliza para recuperar una lista de todas las transferencias de ficheros completadas para el sistema de gestión del proveedor. Esta lista se mantiene en el sistema de gestión del proveedor como registro histórico circular.

A continuación se muestra la signatura de la operación **fileTransferHistoryListGet**:

```
FileTransferHistorySeqType fileTransferHistoryListGet ()
                  raises (AccessDenied);
```

No hay parámetros de entrada.

El valor devuelto es del tipo **FileTransferHistorySeqType** (***tipo de secuencia de historia de transferencia de ficheros***) y proporciona el listado deseado.

9.16.1.7 scheduledFileTransferListGet (*obtener lista de transferencias de ficheros planificadas*)

Esta operación se utiliza para recuperar los nombres de las transferencias de ficheros planificadas existentes, definidas para el sistema de gestión del proveedor.

A continuación se muestra la signatura de la operación **scheduledFileTransferListGet**:

```
ScheduledFileTransferSeqType scheduledFileTransferListGet ()
                  raises (AccessDenied);
```

No hay parámetros de entrada.

El valor devuelto es del tipo **ScheduledFileTransferSeqType** (***tipo de secuencia de transferencia de ficheros planificada***) y proporciona el listado deseado.

9.16.1.8 Excepciones

La excepción **AccessDenied** (***acceso denegado***) se plantea cuando el cliente no tiene autorización de acceso a este objeto de interfaz.

La excepción **CommFailure** (***fallo de comunicaciones***) se plantea cuando hay un fallo de comunicaciones entre el servidor de destino y el sistema de gestión del proveedor.

La excepción **UnknownDestinationServer** (***servidor de destino desconocido***) se plantea cuando el agente de la transferencia no puede acceder al servidor de destino especificado.

La excepción **UnknownRecordSet** (***conjunto de registros desconocido***) se plantea cuando no puede encontrarse el fichero objetivo.

La excepción **UnknownScheduler** (***planificador desconocido***) se plantea cuando el agente de la transferencia no puede acceder al planificador especificado.

La excepción **UnknownTransferProcess** (***proceso de transferencia desconocido***) se plantea cuando el **TransferTrackingObjectId** especificado no puede identificarse.

10 Declaración de conformidad

Las implementaciones que declaren su conformidad con cualquiera de las interfaces definidas en la presente Recomendación deberán observar todas las reglas de comportamiento asociadas a las operaciones de la interfaz así como a las definiciones a las que hace referencia el módulo q834_4:: Q834Common.

Anexo A

Diccionario de datos

La cuadro A.1 proporciona un listado de todos los elementos de datos (tipos de datos) utilizados en la especificación de la interfaz que aparece en esta Recomendación. En este listado se incluye la interpretación, sintaxis y observaciones limitativas de cada información de gestión. Los elementos de datos se enumeran en orden alfabético. Si un tipo de datos, y un segundo tipo de datos construido como secuencia del primero aparecen en la especificación de la interfaz, tan sólo se define el primer elemento de datos, al considerarse que la interpretación del segundo es obvia. En este caso, el nombre de la secuencia está formado por el nombre del elemento de datos original con los caracteres "Seq" insertados antes del final de "Type".

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
`AAL1PMHistoryDataType`	Este elemento de datos proporciona los ítems de este tipo de registro.	`struct`	
`AAL1ProfileType`	Este elemento de datos proporciona los valores de un perfil del tipo AAL1.	`struct`	
`AAL2PMHistoryDataType`	Este elemento de datos proporciona los ítems de este tipo de registro.	`struct`	
`AAL2ProfileType`	Este elemento de datos proporciona los valores de un perfil del tipo AAL2.	`struct`	
`AAL2PVCProfileType`	Este elemento de datos proporciona los valores de un perfil del tipo AAL2PVC.	`struct`	
`AAL5PMHistoryDataType`	Este elemento de datos proporciona los ítems de este tipo de registro.	`struct`	
`AAL5ProfileType`	Este elemento de datos proporciona los valores de un perfil del tipo AAL5.	`struct`	
`AALModeType`	Este elemento de datos indica en qué modo se utiliza el AAL para el VCC de soporte.	`enum`	
`ActivityLevelType`	Especifica el nivel de permiso de acceso concedido a un usuario para su actividad.	`enum`	`monitorOnly`, `allowedTo Execute`, `noAccess`
`ActivityType`	Especifica el tipo de categoría o useractivity (actividad de usuario).	`short`	Definido como constantes en la interfaz `q834_4::Access Control::AccessC ontrolMgr`

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
AdministrationDomain Type	Identificador proporcionado por el OMS u operador durante el registro para indicar el dominio de administración al que pertenece el NE.	UserLabelType	
AdministrativeState Type	Se utiliza para activar (unlock), desactivar (lock), o desconectar (shuttingdown) las funciones de la entidad gestionada asociada.	enum	Se define en Rec. UIT-T X.780.
AggregateATMLoopback ResultType	Especifica el resultado de la prueba de bucle ATM.	struct	
AlarmLogRecordType	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
AlarmStatusSeqType	Este elemento de datos proporciona todos los valores pertinentes de la variable de estado alarm status.	enum	Los valores válidos de enum son AS_UnderRepair, AS_Critical, AS_Major, AS_Minor, AS_AlarmOut standing
AnnouncementType	Este elemento de datos proporciona el anuncio al cliente que descuelga cuando no se ha efectuado llamada alguna.	enum	
APONPMHistoryDataType	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
AppIdType	Este elemento de datos especifica las combinaciones de protocolo utilizadas entre las funciones de interfuncionamiento de la pasarela vocal y el ONT o NT.	enum	
ATMContinuityCheckInfo Type	Especifica la entrada para la comprobación de continuidad ATM.	struct	
ATMLoopbackInfoType	Especifica la información de la prueba de bucle ATM.	struct	
ATMNetworkAccess ProfileType	Perfila un ejemplar del tipo de perfil ATMNetworkAccess.	struct	
ATMOverbookingFactor Type	Este elemento de datos proporciona los valores para el factor de desbordamiento ATM.	struct	
AudioServIndType	Este elemento de datos booleano indica si se transporta o no el servicio de audio, indicando el valor TRUE la presencia de este servicio.	boolean	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
AutoDetectionIndType	Este elemento de datos booleano identifica si está activada, o no, la autodetección de la velocidad de datos.	boolean	
AvailableSysBandwidthSeqType	Éste es un listado de la anchura de banda del sistema disponible por puerto.	Secuencia de tipo PortBandwidthType	
AvailabilityStatusSetType	Este elemento de datos proporciona todos los valores pertinentes de la variable de estado/estado de disponibilidad.	Secuencia de valores de estado de disponibilidad	Se define en Rec. UIT-T X.780.
BackedUpStatusType	Este elemento de datos indica si la entidad gestionada que emite la alarma tiene o no una unidad de respaldo operacional.	boolean	Se define en Rec. UIT-T X.780.
BridgedLANServiceProfileType	Este elemento de datos proporciona los valores para un perfil del género servicio de LAN puenteada.	struct	
BridgePriorityType	Este elemento de datos booleano indica si las funciones de aprendizaje del puente están activadas o no. El valor TRUE significa activada.	short	
BRISignallingType	Este elemento de datos selecciona el formato de señalización a utilizar en la RDSI de velocidad básica.	enum	
BufferedCDVToleranceType	Este elemento de datos representa la duración de los datos de usuario que debe introducir en memoria intermedia la entidad de interfuncionamiento CES para compensar la variación del retardo de células. Esta temporización se efectuará en incrementos de 10 microsegundos.	long long	
CableLengthType	Este elemento de datos proporciona la longitud del cable de pares trenzados entre la interfaz physicalPathTP de tipo "DS1" y el punto de transconexión DSX1 (en su caso).	long long	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
CASType	Este elemento de datos selecciona el formato AAL1 que debe utilizarse. Se aplica únicamente a las interfaces estructuradas. En las interfaces no estructuradas, este valor, de estar presente, debe ponerse al valor por defecto o al básico.	enum	
CASIndType	Este elemento de datos booleano indica si está activada o no la señalización asociada al canal en la conexión; el valor TRUE significa activada.	boolean	
CBRRateType	Este elemento de datos representa la velocidad del servicio CBR soportada por la AAL.	enum	
CCSetUpIdType	Especifica un ID único para el montaje de la prueba en CC.	long long	
CDVTPCREgressType	Tolerancia a la variación del retardo de células – este parámetro es necesario para todas las categorías de servicio. Se aplica al flujo CLP = 0 para ABR y a los flujos CLP = 0 + 1 en los demás casos.	long long	
CDVTPCRIngressType	Tolerancia a la variación del retardo de células – este parámetro es necesario para todas las categorías de servicio. Se aplica al flujo CLP = 0 para ABR y a los flujos CLP = 0 + 1 en los demás casos.	long long	
CellLossIntegrationPeriodType	Este elemento de datos representa el periodo de integración de pérdida de células en milisegundos. Si se pierden células durante este periodo de tiempo, la entidad vcCTPF de interfuncionamiento asociada generará una alarma de descebado de células.	long long	
CESServiceProfileType	Este elemento de datos proporciona los valores para un perfil del tipo CES Service.	struct	
ClockRecoveryType	Este elemento de datos indica si el tipo de recuperación de reloj deriva de la interfaz física.	enum	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
`CMDataIndType`	Este elemento de datos booleano indica si se transportan o no datos en modo circuito por esta conexión; el valor TRUE significa sí.	`boolean`	
`CMMultiplierNumType`	Este elemento de datos proporciona el valor de N en los datos en modo circuito $N \times 64$ kbit/s.	`short`	
`ConformanceDefType`	Indica el tipo de conformidad definido en ATM-Forum TM 4.0.	`enum`	
`ControlStatusSetType`	Este elemento de datos proporciona todos los valores pertinentes de la variable de estado control status.	Secuencia de: `enum`	Se define en Rec. UIT-T X.780.
`CorrelatedNotification Type`	Este elemento de datos lista los números de referencia de otras notificaciones de evento que guarden relación con esta notificación de evento.	Secuencia de: `Notification Identifier Type`	
`CreationModeType`	Este elemento de datos indica cómo se creó el conjunto de registros.	`enum`	Elección entre definido por el operador o ejemplificado como parte de la instalación de la aplicación del sistema de gestión del proveedor.
`CurrentListingType`	Especifica el listado de eventos actuales que pueden sincronizarse entre el sistema de gestión del proveedor y el NE.	`short`	Los valores se especifican como constantes.
`CurrentSizeType`	Este elemento de datos describe el tamaño actual de un conjunto de registros.	`unsigned long long`	En las mismas unidades que `MaxSizeType`
`DataRateType`	Este elemento de datos proporciona la velocidad de datos de la conexión Ethernet. Los valores válidos son 10 Mbit/s o 100 Mbit/s.	`enum`	
`DayOfMonthType`	Especifica el día del mes.	`short`	0 se interpreta como sin especificar.

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
`DayOfWeekType`	Especifica el día de la semana de la planificación.	`enum`	Domingo, Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Sin especificar.
`DCNAddressType`	Proporciona la dirección del NE o servidor del sistema en la red de comunicación de datos del operador. Se utiliza para el encaminamiento de mensajes.	`string`	Normalmente es una dirección IP. Como ejemplos se pueden citar elementos etiquetados como `softwareSource Addr`, `destinationServe rAddr`, `sourceSer verAddr`, `nEDCNAddress` y `newNEDCNAddress`
`DefaultSSCSParameter Profile1PtrType`	Este elemento de datos identifica los valores por defecto para el perfil de servicio de convergencia específico de este servicio asociado a los canales que transportan tráfico de control y del plano de gestión.	`Name`	
`DefaultSSCSParameter Profile2PtrType`	Este elemento de datos identifica los valores por defecto para el perfil de servicio de convergencia específico de este servicio asociado a los canales que transportan secuencias de medios(por ejemplo, canales POTS o RDSI-BA).	`Name`	
`DiagnosticType`	Especifica el tipo de prueba de diagnóstico.	`short`	Específico del proveedor.
`DirectionalityType`	Especifica la direccionalidad de la prueba.	`enum`	Los valores son `Egress`, `Ingress`, `BothDirections`.
`DirectionType`	Especifica la dirección de la longitud de onda óptica asociada a un puerto.	`enum`	Los valores son unidireccional o bidireccional.

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
DownloadStatusSeqType	Este elemento de datos proporciona el estado de una actividad de descarga de soporte lógico.	struct compuesta de ID de un objetivo seguido del estado de entrega, distribución, compromiso y activación.	
DS1EncodingType	Este elemento de datos identifica el tipo de codificación de línea DS1.	enum	
DS1FramingType	Este elemento de datos identifica el tipo de alineación de tramas empleado.	enum	
DS1PMHistoryDataType	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
DS1ProfileType	Este elemento de datos proporciona los valores de un perfil del tipo DS1.	struct	
DS3ApplicationType	Este elemento de datos identifica el tipo de señal DS3.	enum	
DS3PMHistoryDataType	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
DS3ProfileType	Este elemento de datos proporciona los valores de un perfil del tipo DS3.	struct	
DS3EncodingType	Este elemento de datos identifica la codificación de línea DS3.	enum	
DTEDCEType	Este elemento de datos indica si el cableado de la interfaz Ethernet es DTE o DCE.	boolean	
DTMFIndType	Este elemento de datos booleano indica si en la conexión se transportan o no cifras marcadas en multifrecuencia bitono; el valor TRUE significa sí.	boolean	
DuplexType	Este elemento de datos indica si se utiliza el modo dúplex (=TRUE) o semidúplex (=FALSE).	boolean	
E1PMHistoryDataType	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
E3PMHistoryDataType	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
EchoCancellationIndType	Este elemento de datos indica la presencia de dispositivos de compensación de eco.	boolean	
ELCPIndType	Este elemento de datos booleano indica si se está utilizando el control de bucle emulado o no.	boolean	
EncapsulationProtocolType	Este elemento de datos identifica el protocolo de encapsulación utilizado para puentear LAN sobre ATM.	short	
EncProfileIndexType	Este elemento de datos indica el perfil específico de codificación predefinida utilizado.	short	
EncSrcTypeType	Este elemento de datos indica el origen del formato del perfil de codificación. Entre los valores válidos se encuentran "ITU-T" y "ATM Forum".	enum	
EndPointType	Especifica las características de un punto extremo de una conexión.	struct	
EquipmentHolderAddressType	Este elemento de datos identifica la posición del equipo de un paquete de circuitos.	struct entre cuyos componentes se encuentran el número de estante (corta) y el número de puerto (corta)	
EquipmentHolderFType	Esta estructura de datos identifica el elemento equipmentHolderF nombrado en el evento autodescubrimiento.	struct que lista los valores de atributo correspondientes a la entidad gestionada equipmentHolderF	
EquipmentType	Este elemento de datos identifica el elemento inventario nombrado en el evento autodescubrimiento.	short	
EthernetPMHistoryDataType	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
EthernetProfileType	Este elemento de datos proporciona los valores correspondientes a un perfil del tipo Ethernet.	struct	
ExternalTimeType	Este elemento de datos establece la hora local a la que han de asociarse los recursos de red.	GeneralizedTimeType	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
FaxDemodIndType	Este elemento de datos booleano indica si está presente o no una demodulación FAX; el valor TRUE significa sí.	boolean	
FilenameType	Identifica el nombre completo de la ruta de acceso de un fichero. Este fichero puede contener datos de configuración cargados desde un NE, la posición para transferir un fichero de registros o la posición del soporte lógico genérico o del parche de soporte lógico de un NE.	string	
FileTransferHistory Type	Este elemento de datos proporciona un registro relativo a una transferencia de ficheros que siguen anotados en el registro histórico del sistema de gestión del proveedor.	struct	
FilterType	Este elemento de datos se define mediante CosNotifyFilter::Filter.		
FMDataIndType	Este elemento de datos booleano indica si en esta conexión se transportan datos en modo trama, o no; el valor TRUE significa sí.	boolean	
FMMaxFrameLenType	Este elemento de datos proporciona la longitud máxima de una unidad de datos en modo trama.	long long	
ForwardDelayType	Este elemento de datos proporciona el tiempo (en centésimas de segundo) que el puente de la tarjeta Ethernet del ONT (como miembro de la comunidad de todos los puentes de la red de área local puenteada) retiene un paquete antes de entregarlo.	short	
ForwardErrorCorrection Type	Este elemento de datos indica el método de FEC.	enum	
FullActionType	Especifica el comportamiento del conjunto de registros cuando alcanza su tamaño máximo.	enum	Escoger entre **wrap** (eliminar lo más antiguo) y **halt** (detener).
GeneralizedTimeType	Proporciona una métrica normalizada del tiempo. Se utiliza para evitar ambigüedades que afecten a zonas horarias.	string	Sólo se permite un formato YYYYMMDDHHMMSS.fffZ (es decir, hora del Meridiano de Greenwich).

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
HelloTimeType	Este elemento de datos proporciona el intervalo horario (en centésimas de segundo) entre paquetes de saludo (*hello*). Se trata del intervalo de tiempo, en centésimas de segundo, en el que un puente anuncia su presencia mientras es raíz o intenta llegar a serlo.	short	
HistoryDataType	Identifica un tipo de registro compuesto de estadísticas de supervisión de la calidad de funcionamiento.	RecordKind Type	
HourlyDailyWeekly MonthlyIndType	Especifica si la planificación tiene un ciclo temporal horario, diario, semanal o mensual.	enum	Horario, diario, semanal o mensual.
IDLCCallProcessing ProfileType	Este elemento de datos proporciona los valores de un perfil del tipo de procesamiento de llamada IDLC.	struct	
JitterBufferMaxType	Este elemento de datos proporciona la profundidad máxima de la memoria intermedia antifluctuación de fase asociada a este servicio. Las unidades se expresan en milisegundos.	long long	
JitterTargetType	Este elemento de datos proporciona el valor objetivo de la memoria intermedia antifluctuación de fase. El sistema intentará mantener la memoria intermedia antifluctuación de fase en su valor objetivo. Las unidades se expresan en milisegundos.	long long	
LANType	Este elemento de datos proporciona información sobre el tipo de LAN utilizado, por ejemplo, Ethernet, token-ring, etc.	short	
LESProfileType	Este elemento de datos proporciona los valores correspondientes a un perfil del tipo LES.	struct	
LocalMaxNumVCC SupportedType	Este elemento de datos identifica el número de VCC que puede soportar el NE ATM en este lado de la interfaz.	long long	
LocalMaxNumVCIBitsType	Este elemento de datos identifica el número máximo de bits asignados del subcampo VCI que puede soportar el NE FSAN en este lado de la interfaz.	short	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
LocalMaxNumVPC SupportedType	Este elemento de datos identifica el número de VPC que puede soportar el OLT en este lado de la interfaz.	long long	
LocalMaxNumVPIBitsType	Este elemento de datos identifica el número máximo de bits asignados del subcampo VPI que puede soportar el NE FSAN en este lado de la interfaz.	short	
LoopbackCodeType	Este elemento de datos identifica el tipo de código de bucle dentro de banda soportado.	enum	
LoopbackInfoType	Este elemento de datos especifica el tipo de bucle, su direccionalidad y posición.	struct	
LoopbackInfoSeqType	Proporciona información relativa a una de las condiciones de bucle en los recursos de la red.	struct	
LoopbackLocationIdSeq Type	Especifica el Id de posición única para un ctp.	Secuencia de octetos de longitud 16	
LoopbackLocCodeType	Este elemento de datos proporciona el código que identifica células entrantes del bucle OAM de la capa ATM que han de pasar por el bucle de esta UNI.	string	
LoopbackTestType	Este elemento de datos identifica el tipo de configuración de prueba del bucle que se utiliza o se va a utilizar.	unsigned short	Los valores posibles se definen en la interfaz PhysicalLayerLoo pback de Q834: : Common
LoopbackTrackingObject IdType	Este elemento de datos identifica la condición del bucle actual.	Tracking ObjectIdType	
MACBridgePortPMHistory DataType	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
MACBridgePMHistoryData Type	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
MACBridgeService ProfileType	Este elemento de datos proporciona los valores de un perfil del tipo MACBridgeService.	struct	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
`ManagedEntityIdType`	Este elemento de datos proporciona un nombre único para una entidad gestionada. El nombre incluye una indicación de si se trata de un nombre de objeto gestionado de textura fina, de un nombre de objeto de fachada o simplemente hace referencia a una estructura de datos.	`struct` de un `enum` y un `RDNType`	
`MaxAgeType`	Este elemento de datos indica la vida máxima (en segundos) de una entrada en el listado del árbol de expansión. Indica la vida máxima de la información del protocolo recibido antes de ser descartada y se expresa en segundos.	`short`	
`MaxBSEgressType`	Máximo tamaño de ráfaga – este parámetro es necesario para el tráfico VBR en tiempo real y diferido, y para tráfico GFR. Se aplica al flujo de tráfico CLP = 0 + 1 para VBR.1, GFR.1 y GFR.2, y al flujo de tráfico CLP = 0 para VBR.2 y VBR.3.	`long long`	
`MaxBSIngressType`	Máximo tamaño de ráfaga – este parámetro es necesario para el tráfico VBR en tiempo real y diferido, y para tráfico GFR. Se aplica al flujo de tráfico CLP = 0 + 1 para VBR.1, GFR.1 y GFR.2 y al flujo de tráfico CLP = 0 para VBR.2 y VBR.3.	`long long`	
`MaxCPCSSDUSizeType`	Este elemento de datos multivalor representa el máximo tamaño de CPCS_SDU que se transmitirá por la conexión tanto en sentido de transmisión entrante (forward) como saliente (backward).	`struct`	
`MaxCPS_SDULengthType`	Este elemento de datos proporciona la máxima longitud admisible de la unidad de datos del servicio de la subcapa de parte común (o CPS SDU) por la conexión ya sea en sentido de transmisión ascendente o descendente.	`long long`	
`MaxFrameSizeType`	Este elemento de datos indica el máximo tamaño de trama admisible para transmitirla por esta interfaz.	`long long`	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
MaximumChanIdType	Este elemento de datos proporciona el valor máximo admisible del Id de canal en la conexión.	short	
MaxNumChannelsType	Este elemento de datos proporciona el máximo número de canales que puede transportar el camino VC asociado al vcCTP de interfuncionamiento.	short	
MaxNumCIDsType	Este elemento de datos especifica el máximo número de canales del VCC que pueden estar activos.	short	
MaxPacketLengthType	Este elemento de datos especifica la longitud máxima de los paquetes.	long long	
MaxSizeType	Especifica el tamaño máximo del conjunto de registros.	unsigned long long	En unidades a determinar por mutuo acuerdo entre proveedor y operador.
MaxSSSARSDULengthType	Este elemento de datos proporciona la longitud máxima admisible para una SSSAR-SDU de la subcapa de convergencia específica del servicio de reensamblado y segmentación.	long long	
MFR1IndType	Este elemento de datos booleano indica si en la conexión se transportan o no cifras marcadas en multifrecuencia R1; el valor TRUE significa sí.	boolean	
MFR2IndType	Este elemento de datos booleano indica si en la conexión se transportan, o no, cifras marcadas en multifrecuencia R2; el valor TRUE significa sí.	boolean	
MFSEgressType	Tamaño máximo de trama – este parámetro sólo es necesario para el tráfico GFR.	long long	
MFSIngressType	Tamaño máximo de trama – este parámetro sólo es necesario para el tráfico GFR.	long long	
MinimumChanIdType	Este elemento de datos proporciona un listado de los parámetros de calidad de funcionamiento.	short	
MonitoredParameterSeq Type	Este elemento de datos identifica un parámetro supervisado.	string	Los parámetros supervisados se definen en Q834Common:: Monitored Parameter

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
MonitoringKindType	Este elemento identifica el tipo de supervisión de la calidad de funcionamiento especificada.	string	
MonitoringPoint ThresholdsType	Este elemento de datos proporciona un listado de ejemplares de punto de supervisión y los datos umbral asociados para un recurso de red específico.	Secuencia de: struct	
NameType	Este elemento de datos proporciona un nombre CORBA para un objeto perfil.	CosNaming: : Name	
NEFSANType	Esta estructura de datos identifica el elemento de inventario de equipo genérico nombrado en el evento autodescubrimiento .	struct con la lista de los valores de atributos para la entidad gestionada NEFSAN.	
NEKindType	Este elemento de datos identifica el tipo de elementos de red que puede construirse.	enum	A escoger entre BPONOLT, BPONONT, BPONONU y BPONNT. (Nota)
NotificationIdentifier Type	Identifica unívocamente la notificación.	long long	
NTType	Esta estructura de datos identifica el elemento de inventario NT nombrado en el evento autodescubrimiento.	struct que lista los valores de atributo para la entidad gestionada NT.	
OLTType	Esta estructura de datos identifica el elemento de inventario OLT nombrado en el evento autodescubrimiento.	struct que lista los valores de atributo para la entidad gestionada OLT.	
ONTType	Esta estructura de datos identifica el elemento de inventario ONT nombrado en el evento autodescubrimiento .	struct que lista los valores de atributo para la entidad gestionada ONT.	
ONUType	Esta estructura de datos identifica el elemento de inventario ONU nombrado en el evento autodescubrimiento.	struct que lista los valores de atributo para la entidad gestionada ONU.	
OperationalStateType	Especifica el estado operacional (activado o desactivado) de la entidad gestionada.	enum	Se define en Rec. UIT-T X.780.

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
OpticalWaveLengthArraySeqType	Especifica la longitud de onda (en nanómetros) y la dirección de una longitud de onda multiplexada en este puerto óptico.	struct	
ParameterSettingSeqType	Este elemento de datos proporciona un parámetro de supervisión junto con sus parámetros de ventana deslizante asociados.	struct	
PartiallyFilledCellsType	Este elemento de datos booleano identifica el número de octetos de cabeza que se utilizan.	long long	
PasswordType	Proporciona una acreditación de seguridad para acceder a la aplicación o servidores del sistema de gestión del proveedor.	string	Una contraseña suministrada con un Id de usuario debe ratificar la passwordPolicy utilizada por el sistema de gestión del proveedor.
PasswordPolicyType	Especifica la política de contraseñas aplicada por el sistema de gestión del proveedor. Se compone de dos elementos: UserLoginPolicyType y SessionPolicyType.	struct	
PCMEncTypeType	Este elemento de datos indica el tipo de codificación MIC. Entre los valores válidos se encuentran "codificación MIC de ley mu" y "codificación MIC de ley alfa".	short	
PCREgressType	Velocidad de cresta de células – Este parámetro es necesario para el tráfico de cualquier categoría de servicio. Se aplica al flujo CLP = 0 para ABR y al flujo CLP = 0 + 1 en los demás casos.	long long	
PCRIngressType	Velocidad de cresta de células – Este parámetro es necesario para el tráfico de cualquier categoría de servicio. Se aplica al flujo CLP = 0 para ABR y al flujo CLP = 0 + 1 en los demás casos.	long long	
PerceivedSeverityType	Se define en la Rec. UIT-T X.780.	enum	
PIDType	Este elemento de datos identifica los valores del tipo de medios que pueden utilizarse en la encapsulación ATM (se definen en RFC 1483).	short	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
PlugInUnitFType	Esta estructura de datos identifica el elemento de inventario PlugInUnitF nombrado en el evento autodescubrimiento.	struct que lista los valores de atributo de la entidad gestionada PlugInUnit Type	
PlugInUnitType	Específico de la implementación, nombre del tipo de paquetes de circuitos proporcionado por el proveedor.	string	
PortBandwidthSeqType	Este elemento de datos proporciona la anchura de banda en el puerto suministrado.	struct	Como ejemplos se pueden citar ReservedBand widthSeqType y AvailableSysBand widthSeqType
POTSSignallingType	Este elemento de datos selecciona el formato de señalización que debe utilizarse en el servicio POTS.	enum	
ProbableCauseType	Datos correspondientes a la causa probable.	unsigned short	Los valores se definen en Q834Common:: ProbableCause y en Rec. UIT-T X.780.
ProceduralStatusSet Type	Proporciona el estado del procedimiento de una actividad inconclusa.	Secuencia de: enum	Se define en Rec. UIT-T X.780.
ProfileInfoType	Este elemento de datos identifica el tipo de perfil y sus valores de atributo.	struct	
ProfileKindType	Este elemento de datos identifica el tipo de perfil nombrado.	unsigned short	Los valores se facilitan en Q834Common
ProtectionParameter Type	Este elemento de datos describe los parámetros de conmutación de protección asociados al grupo de protección de las entidades gestionadas. Entre los parámetros se encuentran la relación de conmutación de protección, los mecanismos de conmutación de protección permitidos, el indicador de reversión y el tiempo de espera para revertir.	struct	
ProtectionUnitType	Este elemento de datos asocia los recursos de red protegidos y protectores.	struct	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
RASTimerType	Este elemento de datos proporciona el tiempo de reensamblaje (en segundos) de la subcapa de convergencia específica del servicio de segmentación y reensamblaje para la Rec. UIT-T I.366.1.	short	
RateControlIndType	Este elemento de datos booleano indica si en la conexión se transporta, o no, el control de velocidad; el valor TRUE significa sí.	boolean	
RecordType	Éste es un elemento almacenado en un conjunto de registros.	any	El valor es una struct basada en el RecordKindType definido en Q834Common::RecordSetType. En un conjunto de registros sólo puede almacenarse un tipo de RecordKind
RecordSetIdType	El ManagedEntityId del conjunto de registros.	ManagedEntityIdType	
RecordSetStatusType	Especifica el estado actual del RecordSet.	struct de currentSizeType, OperationalStateType,MaxSizeType, SizeThresholdType, filterName, AdministrativeStateType, RecordKindType y RecordSetUserLabel	
RecordKindType	Identifica el tipo de registro.	unsigned short	Los valores se definen en Q834Common::RecordSetType
RecordsSeqType	Este elemento de datos proporciona un conjunto de registros agrupados por tipo.	Secuencia de: any	Véase lo anterior.
RemainingPasswordValidity	Especifica la validez de la contraseña en número de días.	long	Valor ≥ 1

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
ReservationIdType	Identificador que relaciona la anchura de banda reservada de una PON o de un sistema OLT con un ejemplar de servicio.	string	
ReservationInfoType	Este elemento de datos proporciona una cuenta completa de la información de reserva y de la información de conexión del servicio asociado.	struct	
ReservedBandwidthSeq Type	Cuantifica la cantidad de anchura de banda asociada al Id de reserva.	Secuencia de: PortBandwidth Type	
ResourceSelfTestInfo Type	Proporciona un diagnóstico de prueba a utilizar en la autocomprobación de recursos.	short	Los valores vienen determinados por la implementación del proveedor.
ResourceSelfTestResult Type	Especifica un resultado de la autocomprobación de recursos.	struct	
ScheduledFileTransfer Type	Identifica una actividad de transferencia de ficheros planificados pendiente para el sistema de gestión del proveedor.	struct	
ScheduledSynchNEType	Este elemento de datos asocia un NE con su planificación de sincronización.	struct	
ScheduledTestNESeqType	Este elemento de datos asocia un NE con una prueba planificada.	struct	
SchedulerType	Identifica un ejemplar de planificador del sistema de gestión del proveedor.	struct	
SCREgressType	Velocidad de células sostenible – este parámetro se aplica a la VBR en tiempo real y diferido. Se aplica a los flujos de tráfico CLP = 0 + 1 para VBR.1 y a los flujos de tráfico CLP = 0 para VBR.2 y VBR.3.	long long	
SCRIngressType	Velocidad de células sostenible – este parámetro se aplica a la VBR en tiempo real y diferido. Se aplica a los flujos de tráfico CLP = 0 + 1 para VBR.1 y a los flujos de tráfico CLP = 0 para VBR.2 y VBR.3.	long long	
SecurityAlarmLogRecord Type	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
SegmentEndpointIndType	Indica si la vpNetworkCTP construida debe considerarse, o no, segmento o punto extremo en la prueba de bucle de células OAM ATM.	enum	Escoger entre segmento, punto extremo o ninguno.
SegmentLengthType	Este elemento de datos proporciona la longitud del segmento correspondiente a la subcapa de convergencia específica del servicio de segmentación y reensamblaje. Varía entre 0 y el valor máximo proporcionado por el elemento de datos MaxCPS_SDULen.	long long	
SerialNumType	Proporciona el ID de soporte físico único utilizado en la actividad de determinación de distancias definida en la Rec. UIT-T Q.983.1.	string	
ServiceAffectingType	Este elemento de datos indica si una avería afecta al servicio o no, cuando pueda determinarse.	enum	
ServiceCategoryType	Indica la categoría de servicio definida en ATM-Forum TM 4.0. Los valores válidos son CBR, rt-VBR, nrt-VBR, UBR, ABR o GFR.	enum	
ServiceCatType	Este elemento de datos indica el tipo de categoría de servicio proporcionado por AAL2. Entre los valores válidos se encuentran "Audio" y "Multirate".	enum	
ServiceInstanceIdType	Etiqueta proporcionada por el operador asociada al sistema de gestión del proveedor con conexiones.	string	
ServiceOutageRecord Type	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
SessionPolicyType	Especifica las reglas que rigen las sesiones de cliente GUI interconectadas con el sistema de gestión del proveedor.	struct	
SizeThresholdType	Especifica el umbral de disparo de alarmas del conjunto de registros.	unsigned short	Este valor se interpreta como un porcentaje entero (0-100 %) de MaxSize

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
SlotAssignmentType	Proporciona una asociación de un intervalo de tiempo a una asignación de configuración.	struct en la que se muestra el número de intervalo de tiempo y el tipo de tarjeta enchufable.	
SoftwareDownload TrackingObjectIdType	Identifica una actividad de descarga de soporte lógico en curso o que no ha podido completarse con éxito.	TrackObjectId Type	
SONETSDHLinePMHistory Data	Proporciona la estructura de registro que detalla los datos de calidad de funcionamiento recogidos en un intervalo de 15 minutos para el punto de supervisión rsTTPF.	struct	
SONETSDHSection AdaptationPMHistory Data	Proporciona la estructura de registro que detalla los datos de calidad de funcionamiento recogidos en un intervalo de 15 minutos para los puntos de supervisión correspondientes a au3CTPF o au4CTPF.	struct	
SONETSDHSectionPathPMH istoryData	Proporciona la estructura de registro que detalla los datos de calidad de funcionamiento recogidos en un intervalo de 15 minutos para los puntos de supervisión correspondientes a msTTPF, vc3CTPF o vc4CTPF.		
SpanningTreeIndType	Este elemento de datos booleano indica si está activado el algoritmo de árbol de expansión, o no. El valor TRUE significa que está activado.	boolean	
SpecificProblems	Este elemento de datos proporciona un listado de los valores de código de todos los problemas específicos asociados a una avería y alarma.	Secuencia de: short	Valores definidos por el proveedor.
SSADTIndType	Este elemento de datos booleano indica si se ha seleccionado o no el mecanismo de transferencia de datos garantizados. El valor TRUE indica que se ha seleccionado.	boolean	
SSCSParameterProfile1 Type	Este elemento de datos proporciona los valores para un perfil del tipo perfil 1 del parámetro SSCS.	struct	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
SSCSParameterProfile2 Type	Este elemento de datos proporciona los valores para un perfil del tipo perfil 2 del parámetro SSCS.	struct	
SSCSTypeType	Este elemento de datos identifica el tipo de SSCS para la AAL.	enum	
SSTEDIndType	Este elemento de datos booleano indica si se han seleccionado o no los mecanismos de detección de errores de transmisión. El valor TRUE indica que se han seleccionado.	boolean	
StateChangeDefinition Type	Este elemento de datos proporciona un listado de todas las modificaciones de las variables de estado y situación generadas por una avería que dispara una alarma.	Attribute ChangeSetType	Procede de Rec. UIT-T X.780.
StatusAttributeType	Par que muestra el StatusValueType y el porcentaje de compleción de la copia de seguridad o restauración de los datos de configuración asociados a un NE.	struct	
StatusValueType	Proporciona una selección de valores entre ProceduralStatusSetType y OtherStatus que indican aspectos de compleción.	union	
StructuredDataTransfer Type	Este elemento de datos booleano indica si se ha configurado la transferencia de datos estructurados (SDT) en la AAL.	boolean	
SubType	Este elemento de datos identifica el subtipo de AAL.	enum	
SupplyPowerIndType	Este elemento de datos indica si el puerto asociado proporciona, o no, alimentación.	boolean	
SWPValueType	Muestra un punto de supervisión y sus parámetros de ventana deslizante asociados.	Secuencia de: struct	
SynchChangeIndType	Este elemento de datos booleano indica si en la conexión se transporta o no la sincronización del cambio de funcionamiento SSCS. El valor TRUE indica que sí.	boolean	
SystemTimingType	Especifica las fuentes de reloj primaria y secundaria para el NE.	struct de struct	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
T303Type	Este elemento de datos define la longitud máxima del periodo en milisegundos que el ONT ha de esperar la respuesta al mensaje "SETUP" que aparece en la capa 3 para el TMC o el CSC.	enum	
T396Type	Este elemento de datos especifica el periodo de tiempo que el ONT ha de esperar la respuesta de un mensaje "SETUP" tras la expiración inicial del temporizador T303.	enum	
TargetType	Este elemento de datos proporciona un listado de recursos de red.	Secuencia de: struct de ManagedEntity IdType y string.	Permite la selección en el sistema, NE, tipo de unidad enchufable o nivel de ranura específico.
TargetActivityType	Proporciona una asocición de activityLevel, de activityType y dominio administrativo a asociar a un usuario o grupo de usuarios.	struct	
TCAdaptionProtocol MonitoringPMHistory Data	Proporciona la estructura del registro que detalla los datos de calidad de funcionamiento recogidos en un intervalo de 15 minutos para los puntos de supervisión correspondientes a tcAdaptorF.		
TestHistoryType	Este elemento de datos muestra información archivada correspondiente a una actividad de prueba completada para el sistema de gestión del proveedor.	struct	
TestIterationNumType	Este elemento de datos especifica el número de veces que ha de repetirse una prueba.	short	
TestTrackingObjectId Type	Este elemento de datos identifica una prueba que se está realizando o que está pendiente en el sistema de gestión del proveedor.	Tracking ObjectId	
ThresholdDataProfile Type	Este elemento de datos es un listado de pares: un ejemplar thresholdDataProfile y un tipo de punto de supervisión.	Secuencia de: struct	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
ThresholdDataType	Este elemento de datos lista los nombres de parámetros de calidad de funcionamiento y un valor umbral para cada uno de ellos.	struct	
ThresholdInfoType	Este elemento de datos suministra información para un evento de alarma de calidad de servicio y la identidad del parámetro de calidad de funcionamiento, su valor observado y los valores superior e inferior que definen el intervalo de valores umbral.	struct	
ThresholdsList	Este elemento de datos lista las relaciones entre los tipos de punto de terminación de supervisión y sus perfiles de datos umbral asociados.	Secuencia de: struct cuyo primer componente indica el tipo de punto de supervisión y el segundo proporciona una referencia a thresholdData profile.	
ThresholdsType	Este elemento de datos proporciona una asociación de tipos de punto de supervisión asociados a su nombre de perfil de datos umbral.	struct	
TimerCULengthType	Este elemento de datos proporciona el valor para el temporizador de uso combinado Timer_CU.	long long	
TimingReferenceType	Este elemento de datos define cómo se obtiene la temporización interna.	enum	
TotalEgressBandwidth Type	Este elemento de datos identifica la cantidad total de anchura de banda de salida para una interfaz ATM.	long long	
TotalIngressBandwidth Type	Este elemento de datos identifica la cantidad total de anchura de banda de entrada para una interfaz ATM.	long long	
TrackingObjectIdType	Este elemento de datos contribuye a identificar el estado de una actividad solicitada cuyo tiempo de compleción tarda más de varios segundos.	unsigned long	Como ejemplos se pueden citar TransferTracking ObjectIdType, SoftwareDownload TrackingObject Type y TestTracking ObjectIdType.

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
TrafficDescriptor ProfileType	Este elemento de datos proporciona los valores de un perfil del tipo Descriptor de tráfico.	struct	
TransferTrackingObject IdType	Identifica una actividad de transferencia de fichero pendiente, en el sistema de gestión del proveedor.	string	
TriggerTimeMatrixType	Especifica tiempos de activación asociados con un planificador de periodos.	struct	
UNIProfileType	Este elemento de datos proporciona los valores de un perfil del tipo UNI.	struct	
UPCNPCDisagreementPM HistoryDataType	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
UPCNPCIndicatorType	Este elemento de datos booleano determina si se vigilan, o no, todas las conexiones en la interfaz.	boolean	
UsageStateType	Este elemento de datos enumera el valor de la variable de estado de utilización.	enum	Los valores de enum válidos son reposo, activo, ocupado. Se definen en Rec. UIT-T X.780.
UserGroupId	Éste es el nombre de un grupo de usuarios proporcionado por el operador al sistema de gestión del proveedor.	UserLabelType	No puede ser la cadena de caracteres vacía.
UserGroupType	Proporciona la etiqueta de usuario del grupo de usuarios, lista a los usuarios miembros del grupo y enumera sus actividades objetivo.	struct de UserGroupId Type, UserIdSeq Type, y Target Activity SeqType	
UserIdType	UserLabel (etiqueta de usuario) asignada a los usuarios del sistema de gestión del proveedor.	UserLabel	
UserLabelType	Proporciona un identificador creado y proporcionado por el operador o por el OMS para asociarlo al recurso gestionado del sistema de gestión del proveedor.	string	
UserLoginPolicyType	Especifica la política de acceso de los usuarios mediante contraseña.	struct	

Cuadro A.1/Q.834.4 – Elementos de datos y definiciones

Nombre del elemento de datos	Definición	Sintaxis	Observaciones
UserLoginPolicy ViolationReasonType	Especifica un motivo de rechazo de la asignación de una contraseña a un usuario.	enum	
UserType	Proporciona el ID de usuario, el grupo al que pertenece el usuario y las actividades objetivo asignadas al usuario.	struct de UserIdType, UserGroupId SeqType, y Target ActivitySeq Type	
VersionType	Proporciona un identificador de versión del soporte físico y soporte lógico del sistema por managedEntityIdType.	struct	
VoicePMHistoryDataType	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
VoiceServicesProfile AAL1Type	Este elemento de datos proporciona los valores de un perfil del tipo servicios vocales utilizando AAL1.	struct	
VoiceServicesProfile AAL2ProfileType	Este elemento de datos proporciona los valores de un perfil del tipo servicios vocales utilizando AAL2.	struct	
VPVCPMHistoryDataType	Este elemento de datos proporciona los ítems de este tipo de registro.	struct	
NOTA – Si se considera el módulo Q834 Build para ser utilizado en la gestión de otros tipos de tecnologías de acceso, deberán ampliarse los valores enumerados o deberá modificarse la sintaxis a corta, identificando mediante constantes los tipos de elementos de red a construir.			

Anexo B

Excepciones

El cuadro B.1 presenta la relación de todas las excepciones posibles y las circunstancias en las que una o más operaciones de esta especificación de interfaz pueden plantearlas.

Cuadro B.1/Q.834.4 – Excepciones

Excepción planteada	Descripción
AccessDenied (acceso denegado)	El sistema no tiene autorización de acceso a este objeto de interfaz.
ActivationCompleted (activación completada)	Indica que la activación de soporte lógico se ha ejecutado, no pudiendo por tanto cancelarse.
ActivationFailure (fallo de activación)	Fallo del proceso de activación de soporte lógico.
ActivityCompleted (actividad completada)	Se ha ejecutado la actividad de soporte lógico y no puede cancelarse.
ActivityInProgress (actividad en curso)	Esta excepción se plantea cuando la actividad de soporte lógico se ha iniciado y no puede cancelarse.
AddressLabelMismatch (incongruencia de etiqueta de dirección)	El NE identificado no tiene la dirección RCD actual suministrada en la petición.
APONLayerFailure (fallo de capa APON)	Ha habido un fallo de determinación del protocolo APON entre el OLT y el nodo subyacente designado.
BackupInProgress (copia de seguridad en curso)	Esta excepción se plantea cuando se emite la petición durante una operación de copia de seguridad.
CannotAssignManagedEntityId (no se puede asignar Id de entidad gestionada)	El sistema de gestión del proveedor no ha conseguido establecer el ManagedEntityId para el OLT, indicando por tanto que el sistema de gestión del proveedor es incapaz de gestionar el OLT.
CannotRetrieveUserLabel (no se puede recuperar la etiqueta del usuario)	El sistema de gestión del proveedor no ha conseguido leer la etiqueta de usuario suministrada al OLT.
CollectionLimitation (recogida restringida)	El sistema de gestión del proveedor no puede recoger datos durante el tiempo y periodo de granularidad especificados debido a restricciones de la implementación.
CollectionPeriodPast (periodo de recogida superado)	La hora final del periodo es inferior o igual a la hora actual.
CommFailure (fallo de comunicación)	Se ha producido un fallo del enlace RCD entre el NE y el sistema de gestión del proveedor.
ConnectionAlreadyExists (la conexión ya existe)	Ya existe una conexión de subred con los mismos puntos extremos.
ConnectionCountExceeded (cómputo de la conexión sobrepasado)	Se ha sobrepasado el número máximo de conexiones del puerto PON u OLT para esta petición de prestación de servicio.

Cuadro B.1/Q.834.4 – Excepciones

Excepción planteada	Descripción
DCNTimeout (fin de temporización de RCD)	El enlace de comunicaciones RCD entre uno de los NE como mínimo y el sistema de gestión del proveedor está tan congestionado que no puede transferirse la información de estado o situación actual dentro del tiempo de sincronización definido en un sistema.
DeniedAccess (denegación de acceso)	El sistema no tiene autorización de acceso al NE.
DuplicateProfileName (duplicar nombre de perfil)	Esta excepción se plantea cuando el nuevo nombre de perfil duplica un nombre de perfil existente.
DuplicateSerialNumber (número de serie duplicado)	Ya existe otro equipo del mismo tipo con este número de serie.
DuplicateUserGroupId (Id de grupo de usuarios duplicado)	El id ya se utiliza en otro grupo de usuarios.
DuplicateUserId (Id de usuario duplicado)	El perfil de control de acceso ya se ha establecido para este id de usuario.
DuplicateUserLabel (etiqueta de usuario duplicada)	La etiqueta de usuario proporcionada en la petición se ha utilizado para etiquetar otra entidad gestionada al mismo tiempo.
EquipmentFailure (fallo del equipo)	El NE tiene actualmente una avería que impide completar la transacción solicitada.
HWServicesMismatch (desadaptación de servicios de soporte físico)	El NE de sustitución no puede ejecutar los servicios prestados.
InstallationFailure (fallo de instalación)	Fallo del proceso de instalación de soporte lógico.
InsufficientBW (anchura de banda insuficiente)	El algoritmo CAC indica que el servicio solicitado requiere demasiada anchura de banda para el OLT.
InsufficientMemory (memoria insuficiente)	No hay memoria suficiente en el NE para cargar la unidad de soporte lógico.
InsufficientPONBW (anchura de banda de PON insuficiente)	El ONT o la ONU no pueden añadirse debido a que no hay anchura de banda suficiente en el APONLink.
InterfaceSpeedNotChangeable (no es posible modificar la velocidad de la interfaz)	El puerto físico no puede soportar la nueva velocidad de interfaz o la velocidad no es configurable.
IntervalCountTooLarge (cómputo del intervalo demasiado grande)	Esta excepción se plantea cuando el intervalo solicitado sobrepasa el máximo soportado por el sistema de gestión del proveedor. Esta excepción indica el máximo permitido de intervalos de supervisión que soporta el sistema de gestión del proveedor.
InvalidDCNAddress (dirección RCD no válida)	La dirección RCD especificada no es válida.
InvalidEquipmentCode (código de equipo no válido)	El código de equipo no se ajusta a la sintaxis.
InvalidExternalTime (hora externa no válida)	La hora externa especificada no es válida.

Cuadro B.1/Q.834.4 – Excepciones

Excepción planteada	Descripción
InvalidLocationId (Id de posición no válido)	El LLID especificado no es válido.
InvalidPort (puerto no válido)	El puerto PON especificado no es válido.
InvalidProtectionScheme (esquema de protección no válido)	El recurso de red no soporta los parámetros de protección especificados en el contexto con el listado de puertos o bien las unidades de protección son puertos cuyos trayectos físicos son de características diferentes.
InvalidScheduler (planificador no válido)	Los valores de los parámetros del planificador son ajenos al ámbito definido.
InvalidSerialNumSyntax (sintaxis del número de serie no válida)	La sintaxis del número de serie proporcionado no es congruente con las reglas de definición.
InvalidSlotAssignmentList (lista de asignaciones de ranuras no válida)	Las reglas previstas para el suministro de ranuras son violadas por la asignación de ranuras proporcionada.
InvalidSoftwareTrackingObject (objeto de rastreo de soporte lógico no válido)	El objeto de rastreo de soporte lógico de referencia no es el más reciente asociado con la instalación de soporte lógico en el NE.
InvalidStartTime (hora de comienzo no válida)	La hora de comienzo es incongruente con la hora actual, la matriz horaria de activación actual o la nueva hora de detención.
InvalidStopTime (hora de detención no válida)	La nueva hora de detención es incongruente con la matriz horaria de activación actual, la hora actual o con la nueva hora de comienzo.
InvalidTestOperations (operaciones de prueba no válidas)	La operación de autocomprobación solicitada no es válida.
InvalidTimeoutPeriod (periodo de temporización no válido)	El periodo de temporización designado viola la definición de valores válidos.
InvalidTrigger (activador no válido)	El activador especificado tiene valores que el planificador no puede interpretar.
InvalidUserLabelSyntax (sintaxis de etiqueta de usuario no válida)	La UserLabel especificada no cumple las reglas establecidas para userLabel.
LockedAlready (ya bloqueado)	El estado administrativo de la entidad gestionada nombrada ya está bloqueado y no ha realizado su función normal.
MaxSubtendingNodesExceeded (sobrepasado el máximo número de nodos subyacentes)	El número máximo proyectado de nodos subyacentes para la interfaz PON identificada se ha superado con esta petición de prestación de servicio o reserva.
NoResponse (sin respuesta)	El ONT o la ONU no pudieron añadirse y el fallo no obedece a un problema detectado con la sintaxis del número de serie, ni al protocolo de la capa APON ni a la existencia de etiquetas de usuario duplicadas o no válidas.
NoSuchRecords (registros no encontrados)	Ninguno de los registros de los conjuntos de registros designados cumple los criterios de selección.

Cuadro B.1/Q.834.4 – Excepciones

Excepción planteada	Descripción
NoSynchInProgress (no hay sincronización alguna en curso)	Esta excepción se plantea cuando no hay ningún proceso de sincronización en curso.
NotAvailableForTest (no disponible para la prueba)	El CTP especificado no está disponible para esta prueba.
ParameterViolation (violación de parámetros)	Esta excepción se plantea cuando los parámetros del punto extremo son incongruentes con las características de protocolo del puerto, o cuando los valores están fuera de rango o son duplicados no válidos.
ProfileInUse (perfil en uso)	Esta excepción se plantea cuando el perfil no puede suprimirse por estar siendo utilizado para caracterizar entidades gestionadas dentro del campo de gestión del sistema de gestión del proveedor.
ProfileSuspended (perfil suspendido)	El OMS u operador ha suspendido el uso de los perfiles nombrados en la invocación dentro del sistema de gestión del proveedor.
RecordSetExists (ya existe el conjunto de registros)	El conjunto de registros definido por los parámetros de la petición de creación ya existe en el sistema de gestión del proveedor.
RemainingContainedManagedEntities (quedan entidades gestionadas contenidas)	Hay paquetes de circuitos o bastidores de equipos pendientes de suprimir.
RemainingReservations (reservas pendientes)	No puede suprimirse el nodo por existir todavía reservas de recursos.
RemainingSubnetworkConnections (conexiones de subred pendientes)	El nodo de la unidad enchufable no puede suprimirse por quedar conexiones de subred pendientes.
ScheduleInUse (planificación utilizándose)	Quedan actividades planificadas del planificador nombrado.
SlotAlreadyAssigned (ranura ya asignada)	La ranura solicitada ya está ocupada.
SoftwareLoadHardwareMismatch (desadaptación entre el soporte físico y la carga del soporte lógico)	Los datos anteriores de configuración del NE no pudieron descargarse al NE porque se introdujeron modificaciones en el soporte físico del NE que provocaron la imposibilidad.
SoftwareLoadHWMismatch (desadaptación entre el soporte físico y la carga del soporte lógico)	El soporte lógico designado no pudo cargarse en el soporte físico del equipo ya que la versión de éste no puede aceptar la carga del soporte lógico.
SoftwareNotYetInstalled (soporte lógico pendiente de instalar)	El soporte lógico no puede activarse por no haber sido instalado aún.
SoftwareTrackingObjectInUse (objeto de rastreo de soporte lógico utilizado)	Aún quedan actividades del soporte lógico pendientes rastreadas por este objeto y no puede suprimirse.
SourceUnreachable (fuente inaccesible)	El OLT no ha podido acceder al servidor que tiene la carga de soporte lógico que ha de descargarse.

Cuadro B.1/Q.834.4 – Excepciones

Excepción planteada	Descripción
SynchNotScheduled (sincronización no planificada)	Esta excepción se plantea cuando la modificación de la planificación hace referencia a un NE para la que no se ha planificado una sincronización anteriormente.
Timeout (alcanzado el límite de temporización)	La duración del proceso ha alcanzado un límite de temporización definido por el sistema antes de poder completarse.
TooManyNEs (demasiados NE)	El sistema de gestión del proveedor no puede gestionar ningún OLT adicional.
TooManyRecords (demasiados registros)	El número de registros seleccionado para recuperación provoca una respuesta a la petición que supera un tamaño predeterminado.
UncontrolledTestInProgress (no puede cancelarse la autocomprobación)	Debido a que hay una prueba incontrolada.
UnknownBackupProcess (proceso de copia de seguridad desconocido)	El objeto de seguimiento de transferencia nombrado que identifica el proceso de transferencia de ficheros es desconocido para el sistema de gestión del proveedor.
UnknownConnection (conexión desconocida)	La conexión de subred es desconocida para el sistema de gestión del proveedor.
UnknownDestinationServer (servidor de destino desconocido)	El agente de la transferencia no puede acceder al servidor de destino identificado.
UnknownHistoryDataType (tipo de datos históricos desconocido)	El tipo de datos históricos es desconocido para el sistema de gestión del proveedor.
UnknownManagedEntity (entidad de gestión desconocida)	La entidad gestionada especificada es desconocida para el sistema de gestión del proveedor.
UnknownMonitoringPointTypes (tipos de punto de supervisión desconocidos)	El monitoringPointType (*tipo de punto de supervisión*) es desconocido para el sistema de gestión del proveedor.
UnknownNE (NE desconocido)	El NE identificado es desconocido para el sistema de gestión del proveedor.
UnknownOption (opción desconocida)	El valor dado del archivo administrativeState es "ShuttingDown" (*desconectando*).
UnknownParameters (parámetros desconocidos)	El parámetro supervisado en cuestión es desconocido para el sistema de gestión del proveedor.
UnknownPort (puerto desconocido)	El puerto identificado es desconocido para el sistema de gestión del proveedor.
UnknownProfiles (perfiles desconocidos)	Esta excepción se plantea cuando el nombre del perfil suministrado es desconocido para el sistema de gestión del proveedor y no puede recuperarse del repositorio de objetos de perfil.
UnknownRecordSet (conjunto de registros desconocido)	El conjunto de registros identificado en la petición es desconocido para el sistema de gestión del proveedor.

Cuadro B.1/Q.834.4 – Excepciones

Excepción planteada	Descripción
UnknownReservationId (Id de reserva desconocido)	El sistema de gestión del proveedor no reconoce este Id de reserva.
UnknownRestoreProcess (proceso de restauración desconocido)	El objeto de seguimiento de transferencia nombrado que identifica el proceso de transferencia de ficheros es desconocido para el sistema de gestión del proveedor.
UnknownScheduler (planificador desconocido)	El planificador nombrado es desconocido para el sistema de gestión del proveedor.
UnknownService (servicio desconocido)	El servicio descrito es desconocido para el sistema de gestión del proveedor.
UnknownServiceInstance (ejemplar de servicio desconocido)	El ejemplar de servicio es desconocido para el sistema de gestión del proveedor.
UnknownSlot (ranura desconocida)	La ranura solicitada es desconocida en el NE.
UnknownSoftwareDownloadTrackObject (objeto de seguimiento de descarga de soporte lógico desconocido)	La unidad de soporte lógico nombrada (en referencia al soporte lógico distribuido a un NE) es desconocida para el sistema de gestión del proveedor.
UnknownSoftwareLoad (carga de soporte lógico desconocida)	No es posible encontrar la unidad de soporte lógico especificada.
UnknownSourceServer (servidor de origen desconocido)	El sistema de gestión del proveedor y/o el OLT no pueden comunicarse con el servidor de origen. La dirección RCD es desconocida o el acceso está bloqueado.
UnknownSystemTimingSource (fuente de temporización del sistema desconocida)	La fuente de temporización externa es desconocida para el sistema de gestión del proveedor.
UnknownTargets (objetivos desconocidos)	La lista de actividades objetivo es desconocida para el sistema de gestión del proveedor.
UnknownTest (prueba desconocida)	La prueba especificada por el identificador de objeto de seguimiento de prueba es desconocida para el sistema de gestión del proveedor.
UnknownTransferProcess (proceso de transferencia desconocido)	El estado del proceso de transferencia identificado no pudo comprobarse por ser desconocido para el sistema de gestión del proveedor.
UnknownUserGroupId (Id de grupo de usuarios desconocido)	El grupo de usuarios es desconocido para el sistema de gestión del proveedor.
UnknownUserIds (Id de usuarios desconocidos)	Esta excepción se plantea cuando no se reconoce algún userId.
UnrecognisedVersion (versión no reconocida)	La versión del equipo proporcionada no concuerda con los valores conocidos.
UserGroupNotEmpty (grupo de usuarios no vacío)	Grupo de usuarios no vacío que no puede suprimirse.
UserLoginPolicyViolation (violación de política de acceso mediante contraseña de usuario)	La asignación especificada de una nueva contraseña a un usuario viola la política de acceso por contraseña de usuarios actualmente en vigor para el sistema de gestión del proveedor.

Anexo C

Ficheros IDL

Los siguientes ficheros se salvan como ficheros de texto, cualquier subconjunto de ellos puede compilarse satisfactoriamente mediante cualquier compilador OMG IDL que se ajuste a la especificación CORBA 2.1 o posterior, siempre que el subconjunto incluya Q834Common.idl, y CosNaming.idl, CosNotifyFilter.idl, CosNotification.idl, X780.idl y CosNotifyComm.idl normalizados.[11]

C.1 Q834AccessControl.idl

C.1 Q834AccessControl.idl

```
#ifndef __Q834_4_ACCESSCONTROL_DEFINED
#define __Q834_4_ACCESSCONTROL_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module AccessControl {
// Begin definitions from other idl files

// From Q834Common

     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::AdministrationDomainSeqType
                    AdministrationDomainSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;

#define AccessDenied Q834Common::AccessDenied

// End definitions from other idl files

// Local data types

     struct UserLoginPolicyType {
          short minUserId; // Minimum length of userid
          short minPassword; // Minimum length of password
          short passwordReuse;
          short loginAttempts;
          long passwordValidity;
          boolean alphanumeric;//?should password contain alphanumeric mixture
          boolean specialCharacters;
              //?should password contain special characters
          boolean repeatingCharacters; //?should password contain repeating
          boolean disallowUserId; //disallow username in password
     };

     struct SessionPolicyType {
          short sessionInactiveTime;
```

[11] Véanse las referencias [18] y [19].

```
        short inactiveUserIdDisableTime;
        short multipleActiveLogins;
    };

    struct PasswordPolicyType {
        UserLoginPolicyType userLoginPolicy;
        SessionPolicyType sessionPolicy;
    };


    typedef sequence<UserIdType> UserIdSeqType;

    enum ActivityLevelType {
        monitorOnly,  // read
        allowedToExecute,  // write
        noAccess
    };

    typedef short ActivityType;



    struct TargetActivityType {
        ActivityType type;
        ActivityLevelType activityLevel;
        AdministrationDomainSeqType AdministrationDomainSeq;
    };

    typedef sequence<TargetActivityType> TargetActivitySeqType;

    enum UserLoginPolicyViolationReasonType {
        minUserId,
        minPassword,
        passwordReuse,
        loginAttempts,
        passwordValidity,
        alphanumeric,
        specialCharacters,
        repeatingCharacters,
        disallowUserId
    };



    typedef sequence<UserLoginPolicyViolationReasonType>
UserLoginPolicyViolationReasonSeqType;
        typedef sequence<UserLabelType> UserGroupIdSeqType;

    struct UserType {
        UserIdType userId;
        UserGroupIdSeqType userGroupIdSeq;
        TargetActivitySeqType TargetActivitySeq;
    };

    struct UserGroupType {
        UserLabelType userGroupId;
        UserIdSeqType userIdSeq;
        TargetActivitySeqType TargetActivitySeq;
    };

    typedef sequence<UserType> UserSeqType;
    typedef sequence<UserGroupType> UserGroupSeqType;
```

```
// Local exceptions

exception UnknownUserIds {
     UserIdSeqType userIdSeq;
};
exception DuplicateUserId {};
exception UnknownUserGroupId {};
exception DuplicateUserGroupId {};
exception UnknownTargets {
     TargetActivitySeqType unknownTargetActivities;
};
exception UserGroupNotEmpty {};
exception UserLoginPolicyViolation {
     UserLoginPolicyType userLoginPolicy;
     UserLoginPolicyViolationReasonSeqType reason;
};
// End local definitions

valuetype AccessControlMgrValueType: itut_x780::ManagedObjectValueType {

     public PasswordPolicyType passwordPolicy; // GET
     public UserSeqType userList; // GET
     public UserGroupSeqType userGroupList; // GET
};


interface AccessControlMgr : itut_x780::ManagedObject {

// define the activities
     const short ALL_ACTIVITIES = 0;
     const short ACCESS_CONTROL_MANAGEMENT = 1;
     const short ALARM_EVENT_CONFIGURATION_MANAGEMENT = 2;
     const short SCHEDULE_ACTIVITY = 3;
     const short SOFTWARE_DOWNLOAD = 4;
     const short TEST_CONTROL = 5;
     const short SYNCHRONISE_CURRENT_EVENT_LIST = 6;
     const short SYNCHRONISE_NE = 7;
     const short RANGE_NE = 8;
     const short REGISTER_SYSTEM = 9;
     const short RESERVE_RESOURCES = 10;
     const short PROFILE_MANAGEMENT = 11;
     const short PROVISION_NE = 12;
     const short PROVISION_TELEPHONY_SERVICE = 13;
     const short PROVISION_PACKETISED_DATA_SERVICES = 14;
     const short PROVISION_VIDEO_SERVICE = 15;
     const short PROVISION_LEASED_LINE_SERVICE = 16;
     const short BULK_TRANSFER = 17;
     const short HISTORY_DATA_COLLECTION = 18;
     const short CONTROL_ARCHIVING = 19;
     const short CONTROL_PERFORMANCE_MONITORING = 20;
     const short CONFIGURATION_BACKUP_RESTORE = 21;


// See 9.1.1.1 for the description of the behaviour of this operation

     void setPasswordPolicy(
          in PasswordPolicyType passwordPolicy )
          raises ( AccessDenied);

// See 9.1.1.2 for the description of the behaviour of this operation

     PasswordPolicyType passwordPolicyGet()
          raises (AccessDenied);
```

```
// See 9.1.1.3 for the description of the behaviour of this operation

     UserSeqType userListGet ()
          raises (AccessDenied);

// See 9.1.1.4 for the description of the behaviour of this operation

     UserGroupSeqType userGroupListGet ()
          raises (AccessDenied);

// See 9.1.1.5 for the description of the behaviour of this operation

     UserType userGet (
          in UserIdType userId )
          raises (AccessDenied,
               UnknownUserIds);

// See 9.1.1.6 for the description of the behaviour of this operation

     UserGroupType userGroupGet (
          in UserLabelType userGroupId)
          raises (AccessDenied,
               UnknownUserGroupId);


// See 9.1.1.7 for the description of the behaviour of this operation

     void createUserGroup (
          in UserLabelType userGroupId,
          in TargetActivitySeqType targetAdditions)
          raises (DuplicateUserGroupId,
               UnknownTargets,
               AccessDenied);


// See 9.1.1.8 for the description of the behaviour of this operation

     TargetActivitySeqType modifyUserGroup (
          in UserLabelType userGroupId,
          in TargetActivitySeqType targetAdditions,
          in TargetActivitySeqType targetDeletions)
          raises (UnknownUserGroupId,
               UnknownTargets,
               AccessDenied );


// See 9.1.1.9 for the description of the behaviour of this operation

     void deleteUserGroup (
          in UserLabelType userGroupId)
          raises (AccessDenied,
               UserGroupNotEmpty,
               UnknownUserGroupId );



// See 9.1.1.10 for the description of the behaviour of this operation

     void addUsersToGroup (
          in UserLabelType userGroupId,
          in UserIdSeqType userIdList )
          raises (AccessDenied,
               UnknownUserGroupId); // duplicate users are ignored
```

```
    // See 9.1.1.11 for the description of the behaviour of this operation

        void deleteUsersFromGroup (
            in UserLabelType userGroupId,
            in UserIdSeqType userIdList )
            raises (AccessDenied,
            UnknownUserGroupId,
            UnknownUserIds);


    // See 9.1.1.12 for the description of the behaviour of this operation

        TargetActivitySeqType getPermissionList (
            in UserIdType userId )
            raises (UnknownUserIds,
            AccessDenied) ;


    // See 9.1.1.13 for the description of the behaviour of this operation

        TargetActivitySeqType modifyPermissionList (
            in UserIdType userId,
            in TargetActivitySeqType targetAdditions,
            in TargetActivitySeqType targetDeletions )
            raises (UnknownUserIds,
                UnknownTargets,
                AccessDenied);

    // See 9.1.1.14 for the description of the behaviour of this operation

        void createUser (
            in UserIdType userId,
            in PasswordType password,
            in TargetActivitySeqType targetAdditions )
            raises (DuplicateUserId,
                UnknownTargets,
                AccessDenied,
                UserLoginPolicyViolation);



    // See 9.1.1.15 for the description of the behaviour of this operation

        void deleteUser (
            in UserIdType userId )
            raises (UnknownUserIds,
                AccessDenied);


    // See 9.1.1.16 for the description of the behaviour of this operation

        void resetPassword (
            in UserIdType userId,
            in PasswordType newPassword )
            raises (UnknownUserIds,
            UserLoginPolicyViolation,
            AccessDenied);

    }; // interface AccessControlMgr

}; // module AccessControl
```

```
}; // module q834_4

#endif
```

C.2 Q834Build.idl

```
#ifndef __Q834_4_BUILD_DEFINED
#define __Q834_4_BUILD_DEFINED


#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module Build {

// begin definitions from other idl files

// From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserLabelType UserLabelType;

     typedef Q834Common::SlotAssignmentSeqType SlotAssignmentSeqType;
     typedef Q834Common::SerialNumType SerialNumType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::ExternalTimeType ExternalTimeType;
     typedef Q834Common::SystemTimingType SystemTimingType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::ReservationIdSeqType ReservationIdSeqType;
     typedef Q834Common::AdministrationDomainType AdministrationDomainType;
     typedef Q834Common::VersionType VersionType;
     typedef Q834Common::LoopbackLocationIdSeqType LoopbackLocationIdSeqType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::ServiceCategoryType ServiceCategoryType;
     typedef Q834Common::ConformanceDefType ConformanceDefType;
     typedef Q834Common::ATMOverbookingFactorType ATMOverbookingFactorType;


#define AccessDenied Q834Common::AccessDenied
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define InvalidSerialNumSyntax Q834Common::InvalidSerialNumSyntax
#define UnknownProfiles Q834Common::UnknownProfiles
#define InvalidUserLabelSyntax Q834Common::InvalidUserLabelSyntax
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define ParameterViolation Q834Common::ParameterViolation
#define ProfileSuspended Q834Common::ProfileSuspended
#define UnknownNE Q834Common::UnknownNE

// End definitions from other idl files

     // Local data types


     enum NEKindType {
          BPONOLT,
          BPONONT,
          BPONONU,
          BPONNT
     };
```

```
     struct ProtectionUnit
     {
          ManagedEntityIdType portId;
          boolean protectingInd; //TRUE = Protecting, FALSE = Protected
     };

     typedef sequence<ProtectionUnit> ProtectionUnitSeqType;

     enum ProtectionRatioType
     {
          oneForOne,
          oneForN, // N > 1
          mForN // M > 1
     };
               enum AllowedProtectionSwitchingType
     {
          automatic,
          manual,
          both
     };

        struct ProtectionParameterType
        {
          ProtectionRatioType protectionRatio;
          AllowedProtectionSwitchingType protectionSwitchingMethod;
          boolean revertiveInd; // TRUE = revertive
          long waitToRestore; // number of milliseconds before // reverting
        };

     enum SegmentEndpointIndType
     {
          segment,
          endpoint,
          none
     };

enum DirectionType
{
               unidirection,
               bidirection
};



struct OpticalWaveLengthArrayType {
               unsigned long wavelength;
               DirectionType direction;
     };
     typedef sequence <OpticalWaveLengthArrayType>
OpticalWaveLengthArraySeqType;


     // Local exceptions

     exception InvalidExternalTime {};
     exception UnrecognisedVersion {};
     exception DuplicateSerialNumber {};
     exception InvalidSlotAssignmentList {};
     exception RemainingContainedManagedEntities {
               ManagedEntityIdSeqType containedManagedEntities ;
     };
```

```
     exception UnknownNE {};
     exception UnknownSystemTimingSource {};
     exception RemainingReservations {
          ReservationIdSeqType remainingReservationList;
     };
     exception InvalidEquipmentCode {};
     exception SlotAlreadyAssigned {};
     exception UnknownSlot {};
     exception RemainingSubnetworkConnections {
          ManagedEntityIdSeqType containedManagedEntities ;
     };
          exception InterfaceSpeedNotChangeable {};
     exception InvalidProtectionScheme {};


     // End local definitions

     valuetype BuilderValueType: itut_x780::ManagedObjectValueType {

          public ManagedEntityIdSeqType createdNodes; // GET
     };

     interface Builder: itut_x780::ManagedObject {


     // See 9.2.1.1 for the description of the behaviour of this operation


ManagedEntityIdType buildNode (
               in NEKindType nEKind,
               in string supplierName,
               in string location,
               in VersionType hWVersion,
               in SerialNumType serialNum,
               in NameSeqType alarmSeverityProfiles,
               in NameSeqType thresholdDataProfiles,
               in SlotAssignmentSeqType slotAssignmentList,
               in ManagedEntityIdType port, // OLT PON port
               in string modelCode,
               in string systemTitle,
               in VersionSeqType softwareVersions,
               in UserLabelType nEUserLabel,
               in ExternalTimeType externalTime,
               in SystemTimingType systemTiming,
               in AdministrationDomainType administrationDomain )
               raises (UnrecognisedVersion,
                    InvalidSerialNumSyntax,
                    DuplicateSerialNumber,
                    UnknownProfiles,
                    UnknownManagedEntity,
                    DuplicateUserLabel,
                    AccessDenied,
                    InvalidExternalTime,
                    UnknownSystemTimingSource,
                    ProfileSuspended);

     // See 9.2.1.2 for the description of the behaviour of this operation

          void assignUserLabelsToNE (
               in SerialNumType serialNum,
               in UserLabelType nEUserLabel,
               in AdministrationDomainType administrationDomain)
               raises (InvalidSerialNumSyntax,
                    DuplicateSerialNumber,
```

```
                    DuplicateUserLabel,
                    AccessDenied);

     // See 9.2.1.3 for the description of the behaviour of this operation


void modifyNode (
               in ManagedEntityIdType managedEntityId,
               in SlotAssignmentSeqType newSlotAssignmentList,
               in NameSeqType newAlarmSeverityProfiles,
               in NameSeqType newThresholdDataProfiles,
               in ManagedEntityIdType port, // OLT PON Port
               in string newModelCode,
               in UserLabelType newNEuserLabel,
               in ExternalTimeType externalTime,
               in AdministrationDomainType administrationDomain )
               raises (UnknownManagedEntity,
                    UnknownNE,
                    InvalidSlotAssignmentList,
                    UnknownProfiles,
                    DuplicateUserLabel,
                    AccessDenied,
                    InvalidExternalTime,
                    ProfileSuspended);

     // See 9.2.1.4 for the description of the behaviour of this operation

          void deleteNode (
               in ManagedEntityIdType managedEntityId )
               raises (UnknownNE,
                    RemainingContainedManagedEntities,
                    AccessDenied,
                    RemainingReservations,
                    RemainingSubnetworkConnections);



     // See 9.2.1.5 for the description of the behaviour of this operation

          void modifyPort (
               in ManagedEntityIdType physicalPathTPId,
               in NameSeqType newAlarmSeverityProfiles,
               in NameSeqType newThresholdDataProfiles,
               in NameSeqType newPortProfiles,
               in string newFrameFormat,
               in AdministrativeStateType administrativeState,
               in OpticalWaveLengthArraySeqType newOpticalWavelengthArray,
               in LoopbackLocationIdSeqType newLoopbackLocationId,
               in unsigned long newInterfaceSpeed,
               in unsigned long aRCTimer)
               raises (UnknownManagedEntity,
                    UnknownProfiles,
                    AccessDenied,
                    InterfaceSpeedNotChangeable,
                    ProfileSuspended);

     // See 9.2.1.6 for the description of the behaviour of this operation

          ManagedEntityIdType buildPlugInUnit (
               in ManagedEntityIdType nEId,
               in NameType alarmSeverityProfile,
               in UserLabelType plugInUnitUserLabel,
               in string modelCode,
               in ManagedEntityIdType equipmentHolder,
```

```
          in AdministrativeStateType administrativeState)
          raises (UnknownNE,
               DuplicateUserLabel,
               AccessDenied,
               UnknownManagedEntity,
               InvalidEquipmentCode,
               SlotAlreadyAssigned,
               UnknownSlot,
               InvalidSlotAssignmentList,
               UnknownProfiles,
               ProfileSuspended);

// See 9.2.1.7 for the description of the behaviour of this operation

     ManagedEntityIdType modifyPlugInUnit (
          in ManagedEntityIdType plugInUnitId,
          in NameType newAlarmSeverityProfile,
          in string newModelCode,
          in ManagedEntityIdType newEquipmentHolder,
          in UserLabelType newPlugInUnitUserLabel,
          in AdministrativeStateType newAdministrativeState)
          raises (UnknownManagedEntity,
               UnknownProfiles,
               AccessDenied,
               InvalidEquipmentCode,
               SlotAlreadyAssigned,
               UnknownSlot,
               InvalidSlotAssignmentList,
               InvalidUserLabelSyntax,
               ProfileSuspended);



// See 9.2.1.8 for the description of the behaviour of this operation

     void deletePlugInUnit (
          in ManagedEntityIdType plugInUnitId )
          raises (UnknownManagedEntity,
               RemainingSubnetworkConnections,
               AccessDenied,
               RemainingReservations);

// See 9.2.1.9 for the description of the behaviour of this operation

     ManagedEntityIdType buildProtectionGrouping(
          in ProtectionParameterType protectionParameters,
          in ProtectionUnitSeqType protectionUnitList)
          raises (InvalidProtectionScheme,
               AccessDenied);

// See 9.2.1.10 for the description of the behaviour of this operation

     void modifyProtectionParameters(
          in ManagedEntityIdType protectionGroupingId,
          in ProtectionParameterType newProtectionParameters)
          raises (UnknownManagedEntity,
               InvalidProtectionScheme,
                  AccessDenied);

// See 9.2.1.11 for the description of the behaviour of this operation

     void modifyProtectionUnitList(
          in ManagedEntityIdType protectionGroupingId,
          in ProtectionUnitSeqType deltaProtectionUnitList,
          in boolean addDeleteInd) // TRUE = add
```

```
          raises (UnknownManagedEntity,
               InvalidProtectionScheme,
                  AccessDenied);

// See 9.2.1.12 for the description of the behaviour of this operation
     void deleteProtectionGrouping(
          in ManagedEntityIdType protectionGroupingId)
          raises (UnknownManagedEntity,
                  AccessDenied);

// See 9.2.1.13 for the description of the behaviour of this operation
     ManagedEntityIdType buildBridge(
          in NameType mACBridgeProfile,
          in ManagedEntityIdType uplinkPort,
          in ManagedEntityIdSeqType uNIPortList)
          raises (UnknownProfiles,
               AccessDenied,
               UnknownManagedEntity,
               ProfileSuspended);

// See 9.2.1.14 for the description of the behaviour of this operation
     void modifyBridgeProfile(
          in ManagedEntityIdType bridgeId,
          in NameType newMACBridgeProfile)
          raises (UnknownProfiles,
               AccessDenied,
               UnknownManagedEntity,
               ProfileSuspended);

// See 9.2.1.15 for the description of the behaviour of this operation
     void modifyBridgePortList(
          in ManagedEntityIdType bridgeId,
          in ManagedEntityIdSeqType deltaUNIPortList,
          in boolean addDeleteInd)
          raises (AccessDenied,
               RemainingSubnetworkConnections,
               UnknownManagedEntity);

// See 9.2.1.16 for the description of the behaviour of this operation
     void deleteBridge(
          in ManagedEntityIdType bridgeId)
          raises (AccessDenied,
               RemainingSubnetworkConnections,
               UnknownManagedEntity);

// See 9.2.1.17 for the description of the behaviour of this operation
     ManagedEntityIdType buildVPNetworkCTP(
          in ManagedEntityIdType port,
          in short vPI,
          in NameType trafficDescriptorProfileName,
          in ATMOverbookingFactorType overbookingFactor,
          in UserLabelType userLabel,
          in SegmentEndpointIndType segmentEndpointInd)
          raises (AccessDenied,
               UnknownManagedEntity,
               UnknownProfiles,
               ParameterViolation,
               ProfileSuspended);

// See 9.2.1.18 for the description of the behaviour of this operation
     void deleteVPNetworkCTP (
          in ManagedEntityIdType vPNetworkCTP)
          raises (AccessDenied,
               RemainingSubnetworkConnections,
```

```
                    UnknownManagedEntity);

     // See 9.2.1.19 for the description of the behaviour of this operation

          ManagedEntityIdSeqType createdNodesGet ()
               raises (AccessDenied);


     }; // interface Builder

}; // module Build

}; // module q834_4

#endif
```

C.3 Q834Common.idl

```
#ifndef __Q834_4_COMMON_DEFINED
#define __Q834_4_COMMON_DEFINED

#include "CosNaming.idl"
#include "CosNotifyFilter.idl"
#include "itut_x780.idl"
#include "TimeBase.idl"


#pragma prefix "itu.Int"

module q834_4 {

module Q834Common {

// Begin definitions from other idl files

// From CosNaming
     typedef CosNaming::Name NameType;

// From CosNotifyFilter
     typedef CosNotifyFilter::Filter FilterType;

// From TimeBase
     typedef TimeBase::UtcT UtcT;

// From X780

     typedef itut_x780::ProbableCauseType ProbableCauseType;
     typedef itut_x780::AdministrativeStateType AdministrativeStateType;
     typedef itut_x780::OperationalStateType OperationalStateType;
     typedef itut_x780::ProceduralStatusSetType ProceduralStatusSetType;
     typedef itut_x780::PerceivedSeverityType PerceivedSeverityType;
     typedef itut_x780::AvailabilityStatusSetType AvailabilityStatusSetType;
     typedef itut_x780::UsageStateType UsageStateType;
     typedef itut_x780::ControlStatusSetType ControlStatusSetType;
     typedef itut_x780::SpecificProblemSetType SpecificProblemSetType;
     typedef itut_x780::BackedUpStatusType BackedUpStatusType;
     typedef itut_x780::AttributeChangeSetType AttributeChangeSetType;
     typedef itut_x780::SecurityAlarmCauseType SecurityAlarmCauseType;

// End definitions from other idl files

// Local data types
```

```
struct NamingComponentType {
     string type;  // managed entity type
     string id;
};

typedef sequence<NamingComponentType> RDNType;
typedef sequence<RDNType> RDNSeqType;
typedef sequence<NameType> NameSeqType;

enum IdType {
     none,
     x780_fineGrained,
     x780_coarseGrained
};

typedef RDNType MEIdType;

struct ManagedEntityIdType {
     IdType id;
     MEIdType mEId;
};

typedef sequence<ManagedEntityIdType> ManagedEntityIdSeqType;
typedef string UserLabelType;
typedef string SerialNumType;
typedef string VersionType;
typedef string PlugInUnitType;
typedef sequence<UserLabelType> UserLabelSeqType;

typedef UserLabelType AdministrationDomainType;
typedef sequence<AdministrationDomainType> AdministrationDomainSeqType;

typedef string DCNAddressType;

typedef unsigned short RecordKindType;


typedef   string    PlugInType;

// SlotAssignmentList
struct SlotAssignmentType {
     short number;
     PlugInType plugIn; // empty string implies that the slot is not
                    // provisioned for any particular type of plug-in unit
};

typedef sequence <SlotAssignmentType> SlotAssignmentSeqType;
typedef string ReservationIdType;
typedef sequence<ReservationIdType> ReservationIdSeqType;
typedef string ServiceInstanceIdType;
typedef sequence<ServiceInstanceIdType> ServiceInstanceIdSeqType;

typedef UtcT GeneralizedTimeType;
typedef GeneralizedTimeType ExternalTimeType;

enum SystemTimingSourceType {
     internalTimingSource, // freerun
     remoteTimingSource, // external
     slaveTimingTerminationSignal
};

typedef ManagedEntityIdType TimingSourceIdType ;
```

```
struct SystemTimingComponentType {
     SystemTimingSourceType type;
     TimingSourceIdType  id;
};

struct SystemTimingType {
     SystemTimingComponentType Primary;
     SystemTimingComponentType Secondary;
};


typedef UserLabelType UserIdType;
typedef string PasswordType;

/*
All zeros indicate a loopback request directed at all connection
points having an LLID in the flow. All ones indicate a loopback
request directed at the endpoint (segment or connection endpoint).
'x6A'H indicates no designated CP for loopback, and therefore no
loopback should be performed. All other values for LLID indicate
a loopback request directed at a specific LLID location.
*/
typedef   sequence<octet,16> LoopbackLocationIdSeqType;

typedef string FilenameType; // specifies the complete path

enum StatusType {
     procedural_status,
     other
};

enum OtherStatusType {
     completed_success,
     completed_failure,
     activityInProgress,
     unknownstatus
};

union StatusValueType switch (StatusType) {
     case procedural_status: ProceduralStatusSetType proceduralStatusSet;
     case other: OtherStatusType otherStatus;
};

struct StatusAttributeType {
     StatusValueType valueOfStatus;
     ManagedEntityIdType ne;
     short percentComplete;
};

typedef sequence<StatusAttributeType> StatusAttributeSeqType;

typedef  unsigned long TrackingObjectIdType;

typedef TrackingObjectIdType TransferTrackingObjectIdType;

typedef any RecordType; // based on recordType (Values defined in
Q834Common::RecordSetType) Need definition of recordType and individual type of
records
typedef sequence<RecordType> RecordSeqType;


typedef unsigned long long NotificationIdentifierType; // will be replaced
                    // by NotifIDType defined in X.780 when that definition
is changed from "long" to "long long"
```

```
     typedef sequence<NotificationIdentifierType> NotificationIdentifierSeqType;


     typedef stringMonitoredParameterType; // Values defined in
Q834Common::MonitoringParameter
     typedef unsigned short MonitoringKindType; // values defined in
Q834Common::PMCategory interface

     struct EndPointType {
          ManagedEntityIdType portId;
          any endPointParameters;
          NameSeqType serviceCharacteristicsProfiles;
     };

     /*
     endPointParameters: service specific parameters, structures to be
     provided as part of implementation. An ATM connection for example
     would have the VPI, VCI parameters.
     */


     enum AlarmStatusComponentType {
          AS_UnderRepair,
          AS_Critical,
          AS_Major,
          AS_Minor,
          AS_AlarmOutstanding
     };

     typedef sequence<AlarmStatusComponentType> AlarmStatusSeqType;

     struct EquipmentHolderAddressType {
          short shelfNumber;
          short slotNumber;
     };

     /*
     If the equipment holder is a shelf, the slot number is 0. If the
     equipment holder is a slot, then both the slot number and shelf numbers
     are greater than or equal to 1.
     */

     /*
     The supplier provides a customization of the DiscoveryEventSupplier
     interface by defining the const string plugInUnit names. It will be up to
     the operator to enforce consistency and non-duplication across multiple
     supplier's solutions.
     */

     typedef sequence<PlugInUnitType> PlugInUnitSeqType;



struct NEFSANType {
          ManagedEntityIdType managedEntityId;
          AdministrativeStateType administrativeState;
          OperationalStateType operationalState;
          GeneralizedTimeType externalTime;
          string locationName;
          string supplierName;
          VersionType hardwareVersion;
          VersionSeqType softwareVersions;
          string serialNumber;
```

```
     NameSeqType alarmSeverityAssignmentProfileNames;
     AlarmStatusSeqType alarmStatusValues;
     NameSeqType thresholdDataNames;
     ManagedEntityIdSeqType supportedByManagedEntityList;
     UserLabelType userLabel;
};
/*
SupportedByManagedEntityList should include the instance of Software
controlling the NE.
*/

/*
Next, structs are defined that form the basis for equipment
inventory information discovered at installation or with equipment
rearrangement actions. Any one of these structs is subsequently
identified as Mestruct in the DiscoveryEventSupplier interface
specification.
*/

struct OLTType {
     NEFSANType nEFSAN;
     ManagedEntityIdSeqType subtendingNEFSANList;
};

struct ONTType {
     NEFSANType nEFSAN;
     ManagedEntityIdType upstreamNEFSAN;
};

struct ONUType {
     NEFSANType nEFSAN;
     ManagedEntityIdType upstreamNEFSAN;
     ManagedEntityIdSeqType subtendingNEFSANList;
};

struct NTType {
     NEFSANType nEFSAN;
     ManagedEntityIdType upstreamNEFSAN;
};

struct EquipmentHolderFType {
     ManagedEntityIdType equipmentHolderFId;
     ManagedEntityIdType containingNEId;
     EquipmentHolderAddressType equipmentHolderAddress;
     boolean slotStatus;
     PlugInUnitSeqType expectedPlugInUnits;
     string SoftwareLoad;
     NameType alarmSeverityAssignmentProfileName;
     AlarmStatusSeqType alarmStatusValues;
     OperationalStateType operationalState;
};

struct PlugInUnitFType {
     ManagedEntityIdType plugInUnitFId;
     ManagedEntityIdType containingNEId;
     EquipmentHolderAddressType containingSlotAddress;
     AdministrativeStateType administrativeState;
     AvailabilityStatusSetType availabilityStatus;
     OperationalStateType operationalState;
     string modelCode;
     string functionCode;
     string supplierName;
     VersionType hardwareVersion;
     string serialNumber;
```

```
        short portCount;
        NameSeqType alarmSeverityAssignmentProfileNames;
        NameSeqType thresholdDataNames;
        UserLabelType circuitPackUserLabel;
        ManagedEntityIdSeqType supportedByManagedEntityList;
    };

    enum ServiceCategoryType {
        CBR,
        UBR,
        RTVBR,
        NRTVBR,
        AdaptiveBR,
        GFR
    };

    enum ConformanceDefType {
        CBR1,
        UBR1,
        UBR2,
        VBR1,
        VBR2,
        VBR3,
        ABR,
        GFR1,
        GFR2
    };

    struct ATMOverbookingFactorType {
        ServiceCategoryType serviceCategory;
        ConformanceDefType conformanceDef;
        short overbookingFactor; // percentage: 100 = no overbooking
    };

    struct AlarmLogRecordType {
        long long recordId;
        ManagedEntityIdType mEId;
        GeneralizedTimeType loggingTime;
        short probableCause;
        PerceivedSeverityType severity;
        GeneralizedTimeType eventTime;
        ManagedEntityIdType backupEntityId;
        boolean backedupStatus;
        boolean serviceAffectingInd;
        ServiceInstanceIdSeqType affectedServices;
        NotificationIdentifierType notificationId;
        NotificationIdentifierSeqType correlatedNotifications;
        string monitoredParameter;
        long long thresholdValue;
        long long observedValue;
        string additionalText;
    };

    struct SecurityAlarmLogRecordType {
        long long recordId;
        ManagedEntityIdType mEId;
        GeneralizedTimeType loggingTime;
        short securityAlarmCause; // values defined in X.780
        GeneralizedTimeType eventTime;
        ManagedEntityIdType securityAlarmDetector;
        ManagedEntityIdType serviceUser;
        ManagedEntityIdType serviceProvider;
        NotificationIdentifierType notificationId;
        NotificationIdentifierSeqType correlatedNotifications;
```

```
     string additionalText;
};

struct ServiceOutageRecordType {
     long long recordId;
     ServiceInstanceIdType affectedService;
     GeneralizedTimeType loggingTime;
     GeneralizedTimeType outageStartTime;
     GeneralizedTimeType outageEndTime;
     NotificationIdentifierSeqType correlatedNotifications;
};

struct AAL1PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long HeaderErrors;
     unsigned long long LostCells;
     unsigned long long CellMisinsertion;
     unsigned long long BufferUnderflows;
     unsigned long long SequenceViolations;
     unsigned long long SDTPtrReframes;
     unsigned long long SDTPtrParityCheckFailures;
};

struct AAL2PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId; unsigned long long CPSInPkts;
     unsigned long long CPSOutPkts;
     unsigned long long BufferUnderflow;
     unsigned long long BufferOverflow;
     unsigned long long ParityErrors;
     unsigned long long SeqNumErrors;
     unsigned long long CPS_OSFMismatchErrors;
     unsigned long long CPS_OSFErrors;
     unsigned long long CPSHECErrors;
     unsigned long long OversizedSDUErrors;
     unsigned long long ReassemblyErrors;
     unsigned long long HECOverlapErrors;
     unsigned long long UUIErrors;
     unsigned long long CIDErrors;
};

struct AAL5PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long SumOfInvalidCSFieldErrors;
     unsigned long long CRCViolations;
     unsigned long long BufferOverflows;
     unsigned long long EncapProtocolErrors;
};

struct APONPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
```

```
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ES;
     unsigned long long FEES;
};

struct ATMTrafficLoadHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long CellsReceived;
     unsigned long long CellsTransmitted;
};

struct DS1PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ESP;
     unsigned long long BESP;
     unsigned long long SESP;
     unsigned long long UASP;
     unsigned long long ESPFE;
     unsigned long long BESPFE;
     unsigned long long SESPFE;
     unsigned long long UASPFE;
};

struct DS3PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ESL;
     unsigned long long SESL;
     unsigned long long CVCPorCVPP;
     unsigned long long ESCPPorESPP;
     unsigned long long SESCPPorSESPP;
     unsigned long long UASCPPorUASPP;
};

struct E1PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ESP;
     unsigned long long BESP;
     unsigned long long SESP;
     unsigned long long UASP;
     unsigned long long ESPFE;
     unsigned long long BESPFE;
     unsigned long long SESPFE;
     unsigned long long UASPFE;
};

struct EthernetHistoryDataType {
     long long recordId;
```

```
    ManagedEntityIdType monitoringPoint;
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;
    unsigned long long SingleCollisionFrame;
    unsigned long long MultipleCollisionFrames;
    unsigned long long SQE;
    unsigned long long DeferredTransmission;
    unsigned long long LateCollision;
    unsigned long long ExcessiveCollision;
    unsigned long long InternalMACTransmitError;
    unsigned long long CarrierSenseError;
    unsigned long long BufferOverflows;
    unsigned long long AlignmentError;
    unsigned long long FrameTooLongs;
    unsigned long long FCSErrors;
    unsigned long long InternalMACReceiveError;
};

struct MACBridgePMHistoryDataType {
    long long recordId;
    ManagedEntityIdType monitoringPoint;
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;
    unsigned long long NumberofSuppressedIntervals;
    unsigned long long BridgeLearningEntryDiscard;
};

struct MACBridgePMPortHistoryDataType {
    long long recordId;
    ManagedEntityIdType monitoringPoint;
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;
    unsigned long long ForwardedFrame;
    unsigned long long DelayExceededDiscard;
    unsigned long long MTUExceededDiscard;
    unsigned long long ReceivedFrame;
    unsigned long long ReceivedAndDiscarded;
};

struct UpcNpcDisagreementPMHistoryDataType {
    long long recordId;
    ManagedEntityIdType monitoringPoint;
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;
    unsigned long long DiscardedCells;
    unsigned long long DiscardedCLP_0Cells;
    unsigned long long TaggedCLP_0Cells;
};

struct VoicePMHistoryDataType {
    long long recordId;
    ManagedEntityIdType monitoringPoint;
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;
    unsigned long long IncomingCallAttempts;
    unsigned long long OutgoingCallAttempts;
    unsigned long long VoicePortBufferOverflows;
    unsigned long long VoicePortBufferUnderflows;
};
```

```
struct VpVcPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long Lost0plus1UserInformationCells;
     unsigned long long Lost0UserInformationCells;
     unsigned long long MisinsertedUserInformationCells;
     unsigned long long Transmitted0plus1UserInformationCells;
     unsigned long long Transmitted0UserInformation;
     unsigned long long ImpairedBlock;
};

struct SONETSDHLinePMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ErroredSecondsP;
     unsigned long long SeverelyErroredSecondsP;
     unsigned long long BackgroundBlockErrorP;
     unsigned long long OutOfFrameSecondsP;
     unsigned long long UnavailableSecondsP;
};

struct SONETSDHSectionPathPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ErroredSecondsP;
     unsigned long long SeverelyErroredSecondsP;
     unsigned long long BackgroundBlockErrorP;
     unsigned long long OutOfFrameSecondsP;
     unsigned long long UnavailableSecondsP;
     unsigned long long FailureCountP;
     unsigned long long ErroredSecondsTypeAP;
     unsigned long long ErroredSecondsTypeBP;
     unsigned long long ErroredSecondsPFE;
     unsigned long long SeverelyErroredSecondsPFE;
     unsigned long long BackgroundBlockErrorPFE;
     unsigned long long OutOfFrameSecondsPFE;
     unsigned long long UnavailableSecondsPFE;
     unsigned long long FailureCountPFE;
     unsigned long long ErroredSecondsTypeAPFE;
     unsigned long long ErroredSecondsTypeBPFE;
};

struct SONETSDHSectionAdaptationPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long PointerJustificationHighCountP;
     unsigned long long PointerJustificationLowCountP;
};

struct TCAdaptationProtocolMonitoringPMHistoryDataType {
     long long recordId;
```

```
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId;
        unsigned long long DiscardedCellsHECViolationP;
        unsigned long long ErroredCellsHECViolationP;
    };

//Common exceptions

    exception AccessDenied {};
    exception CommFailure {};
    exception ConnectionCountExceeded {};
    exception DuplicateUserLabel {};
    exception EquipmentFailure {};
    exception InsufficientBW {};
    exception InsufficientPONBW {
        ManagedEntityIdType ponPORT;
    };
    exception InvalidSerialNumSyntax {};
    exception InvalidUserLabelSyntax {};
    exception MaxSubtendingNodesExceeded {};
    exception Timeout {};
    exception UnknownManagedEntity {
        ManagedEntityIdType managedEntityId;
    };
    exception UnknownNE {
        ManagedEntityIdSeqType unknownNEs;
    };
    exception UnknownPort {};
    exception UnknownProfiles {};
    exception UnknownReservationId {};
    exception UnknownScheduler {};
    exception InvalidScheduler {};
    exception DCNTimeout {};
    exception UnknownDestinationServer {};
    exception UnknownServiceInstance {};
    exception DeniedAccess {};
    exception BackupInProgress {};
    exception InvalidStartTime {};
    exception InvalidStopTime {};
    exception ParameterViolation {};
    exception UnknownRecordSet {};
    exception SynchInProgress {};
    exception ProfileSuspended {
        NameSeqType profilesSuspended;
    };


    module ProbableCauseConst {

        const string moduleName = "q834_4::Q834Common::ProbableCauseConst";

        interface ProbableCause {
        /*
        For X.780 probableCause refer to itut_x780::ProbableCauseConst.
        */

        // Additional ProbableCause values needed for BPON

            const unsigned short LOSS_OF_PATH_POINTER = 1;
            const unsigned short STS_PAYLOAD_LABEL_MISMATCH = 2;
            const unsigned short STS_PATH_UNEQUIPPED = 3;
            const unsigned short ALARM_INDICATION_SIGNAL = 4;
```

```
            const unsigned short REMOTE_FAILURE_INDICATION = 5;
            const unsigned short REMOTE_ALARM_INDICATION = 6;
            const unsigned short ALARM_INDICATION_SIGNAL_PATH = 7;
            const unsigned short
ALARM_INDICATION_SIGNAL_CUSTOMER_INSTALLATION = 8;
            const unsigned short LOSS_OF_SIGNAL_TO_ALL_ONU_ONU = 9;
            const unsigned short LOSS_OF_SIGNAL_ONU_OR_ONT = 10;
            const unsigned short LOSS_OF_ACKNOWLEDGEMENT_ONU_OR_ONT = 11;
            const unsigned short PLOAMCL_ONU_OR_ONT = 12; //Physical Layer OAM
Cell Loss
            const unsigned short LOCD_ONU_OR_ONT = 13; // Loss of Cell
Delineation
            const unsigned short CPHE_ONU_OR_ONT = 14; // Cell Phase Error
            const unsigned short PEE_ONU_OR_ONT = 15;
            const unsigned short RF_ONU_OR_ONT = 16; // Ranging Failure
            const unsigned short BED_ONU_OR_ONT = 17; // Block Error Detection
            const unsigned short SD_ONU_OR_ONT = 18; // Signal Degraded
            const unsigned short REI_ONU_OR_ONT = 19; // Remote Error
Indication
            const unsigned short UM_ONU_OR_ONT = 20; // Unknown Message
            const unsigned short LM_ONU_OR_ONT = 21; // Link Mismatch
            const unsigned short REMOTE_DEFECT_INDICATION = 22; // Signal
Degraded
            const unsigned short MAJOR_POWER_FAILURE = 23;
            const unsigned short
REMOTE_ALARM_INDICATION_FAR_END_CUSTOMER_INSTALLATION = 24;
            const unsigned short LOSS_OF_ATM_CELL_DELINEATION = 25;
            const unsigned short LOW_BATTERY_THRESHOLD = 26;
            const unsigned short DIAGNOSTIC_TEST_FAILURE = 27; // See below
comment
            const unsigned short LOSS_OF_DCN_LINK = 28;
            const unsigned short CELL_STARVATION = 29;
            const unsigned short UNEXPECTED_PLUGIN = 30;
            const unsigned short IMPROPER_CARD_REMOVAL = 31;
            const unsigned short SLOT_CARD_MISMATCH = 32;
            const unsigned short LOS_LAN = 33; // Loss of carrier on Bridge
LAN port
            const unsigned short PERSISTENT_IMPAIRMENT = 34;
        }; // interface ProbableCause
    }; // module ProbableCauseConst

    interface MonitoringParameter {
        /*
        Names for monitored performance and traffic parameters
        */

        const string allParameters = "AllParameters";
        const string aAL1HeaderErrors = "AAL1HeaderErrors";
        const string allTypesCellsDiscarded = "AllTypesCellsDiscarded";
        const string aTMProtocolErrors = "ATMProtocolErrors";
        const string bESFEP = "BESFEP";
        const string bESP = "BESP";
        const string bufferOverflows = "BufferOverflows";
        const string bufferUnderflows = "BufferUnderflows";
        const string cellDelineationAnomalies = "CellDelineationAnomalies";
        const string cellMisinsertion = "CellMisinsertion";
        const string cRCViolations = "CRCViolations";
        const string cVCP = "CVCP";
        const string cVL = "CVL";
        const string cVPP = "CVPP";
        const string cVS = "CVS";
        const string discardedAllTypeCellsduetoNPC =
"DiscardedAllTypeCellsduetoNPC";
```

```
        const string discardedAllTypeCellsduetoUPC =
"DiscardedAllTypeCellsduetoUPC";
        const string discardedPriorityCellsduetoNPC =
"DiscardedPriorityCellsduetoNPC";
        const string discardedPriorityCellsduetoUPC =
"DiscardedPriorityCellsduetoUPC";
        const string encapProtocolErrors = "EncapProtocolErrors";
        const string eSCPP = "ESCPP";
        const string eSPONT = "ESPONT";
        const string eSL = "ESL";
        const string eSP = "ESP";
        const string eSPP = "ESPP";
        const string eSS = "ESS";
        const string excessiveCollisions = "ExcessiveCollisions";
        const string fCSErrors = "FCSErrors";
        const string frameTooLongs = "FrameTooLongs";
        const string hECViolations = "HECViolations";
        const string impairedBlocks = "ImpairedBlocks";
        const string lateCollisions = "LateCollisions";
        const string lostCells = "LostCells";
        const string lostPriorityUserInformationCells =
"LostPriorityUserInformationCells";
        const string lostUserInformationCells = "LostUserInformationCells";
        const string misinsertedUserInformationCells =
"MisinsertedUserInformationCells";
        const string priorityCellsDiscarded = "PriorityCellsDiscarded";
        const string sDTPointerReframes = "SDTPointerReframes";
        const string sDTPointerParityCheckFailures =
"SDTPointerParityCheckFailures";
        const string sequenceViolations = "SequenceViolations";
        const string sESCPP = "SESCPP";
        const string sESPONT = "SESPONT";
        const string sESL = "SESL";
        const string sESP = "SESP";
        const string sESPP = "SESPP";
        const string sESS = "SESS";
        const string sumOfInvalidCSFieldErrors = "SumOfInvalidCSFieldErrors";
        const string uASPONT = "UASPONT";
        const string uASCPP = "UASCPP";
        const string uASP = "UASP";
        const string uASPP = "UASPP";
        const string incomingCallAttempts = "IncomingCallAttempts";
        const string outgoingCallAttempts = "OutgoingCallAttempts";
        const string voicePortBufferOverflows = "VoicePortBufferOverflows";
        const string voicePortBufferUnderflows = "VoicePortBufferUnderflows";
        const string cPSInPkts = "CPSInPkts";
        const string cPSOutPkts = "CPSOutPkts";
        const string bufferUnderflow = "BufferUnderflow";
        const string bufferOverflow = "BufferOverflow";
        const string parityErrors = "ParityErrors";
        const string seqNumErrors = "SeqNumErrors";
        const string cPS_OSFMismatchErrors = "CPS_OSFMismatchErrors";
        const string cPS_OSFErrors = "CPS_OSFErrors";
        const string cPSHECErrors = "CPSHECErrors";
        const string oversizedSDUErrors = "OversizedSDUErrors";
        const string reassemblyErrors = "ReassemblyErrors";
        const string hECOverlapErrors = "HECOverlapErrors";
        const string uUIErrors = "UUIErrors";
        const string cIDErrors = "CIDErrors";

    }; // interface MonitoringParameter

    interface PMCategory {
        const unsigned short DS1_PM = 1;
```

```
        const unsigned short DS3_PM = 2;
        const unsigned short E1_PM = 3;
        const unsigned short E3_PM = 4;
        const unsigned short VP_PM = 5;
        const unsigned short VC_PM = 6;
        const unsigned short ETHERNET_PM = 7;
        const unsigned short TC_ADAPTOR_PM = 8;
        const unsigned short VOICE_PM = 9;
        const unsigned short APON_PM = 10;
        const unsigned short MACBRIDGE_PM = 11;
        const unsigned short MACBRIDGEPORT_PM = 12;
        const unsigned short ATMTRAFFICLOAD_PM = 13;
        const unsigned short AAL1_PM = 14;
        const unsigned short AAL2_PM = 15;
        const unsigned short AAL5_PM = 16;
        const unsigned short UPCNPC_PM = 17;
        const unsigned short DBA_PM = 18;
        const unsigned short SONET_SDH_LINE_PM = 19;
        const unsigned short SONET_SDH_SECTION_PATH_PM = 20;
        const unsigned short SONET_SDH_SECTION_ADAPTATION_PM = 21;
        const unsigned short CALL_STATS_PM = 22;
        const unsigned short VIDEO_PM = 23;

    }; // interface PMCategory

    interface RecordSetType {

    // Beginning of Values for RecordKindType
    // Values 1-99 reserved for HistoryDataType

        const unsigned short DS1PMHISTORYDATA = 1;
        const unsigned short DS3PMHISTORYDATA = 2;
        const unsigned short E1PMHISTORYDATA = 3;
        const unsigned short E3PMHISTORYDATA = 4;
        const unsigned short VPVCPMHISTORYDATA = 5;
        const unsigned short AAL1PMHISTORYDATA = 6;
        const unsigned short AAL2PMHISTORYDATA = 7;
        const unsigned short AAL5PMHISTORYDATA = 8;
        const unsigned short UPCNPCDISAGREEMENTPMHISTORYDATA = 9;
        const unsigned short ETHERNETPMHISTORYDATA = 10;
        const unsigned short VOICEPMHISTORYDATA = 11;
        const unsigned short MACBRIDGEPORTPMHISTORYDATA = 12;
        const unsigned short MACBRIDGEPMHISTORYDATA = 13;
        const unsigned short APONPMHISTORYDATA = 14;
        const unsigned short SONETSDHLINEPMHISTORYDATA = 15;
        const unsigned short SONETSDHSECTIONADAPTATIONPMHISTORYDATA = 16;
        const unsigned short SONETSDHSECTIONPATHPMHISTORYDATA = 17;
        const unsigned short TCADAPTATIONPROTOCOLMONITORINGPMHISTORYDATA = 18;
        const unsigned short ALARMLOGRECORD = 100;
        const unsigned short SECURITYALARMLOGRECORD = 101;
        const unsigned short SERVICEOUTAGERECORD = 102;

    // End of Values for RecordKindType

    }; // interface RecordSetType

interface PhysicalLayerLoopback {

    // Beginning of Values for LoopbackTestType

        const unsigned short LINELOOPBACK = 1;
        const unsigned short PAYLOADLOOPBACK = 2;
        const unsigned short INWARDLOOPBACK = 3;
        const unsigned short DUALLOOPBACK = 4;
```

```
          const unsigned short FACILITYLOOPBACK = 5;
          const unsigned short TERMINALLOOPBACK = 6;

     // End of Values for LoopbackTestType

     }; // interface  PhysicalLayerLoopback

};   // module Q834Common

}; // module q834_4
#endif
```

C.4 Q834ControlArchive.idl

```
#ifndef __Q834_4_CONTROLARCHIVE_DEFINED
#define __Q834_4_CONTROLARCHIVE_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module ControlArchive {

// begin definitions from other idl files

// From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::FilterType FilterType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::OperationalStateType OperationalStateType;
     typedef Q834Common::RecordType RecordType;
     typedef Q834Common::RecordSeqType RecordSeqType;
     typedef Q834Common::UserLabelSeqType UserLabelSeqType;
     typedef Q834Common::RecordKindType RecordKindType;


#define AccessDenied Q834Common::AccessDenied
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define Timeout Q834Common::Timeout
#define UnknownRecordSet Q834Common::UnknownRecordSet

// End definitions from other idl files

// Local data types

     enum FullActionType {
          halt, // indicates that the log should stop writing any more records
if the log is full
          wrap // indicates that the log should delete old records if the log is
full.
     };

     typedef unsigned long long MaxSizeType;
     typedef unsigned short SizeThresholdType; // 0-100%
     typedef unsigned long long CurrentSizeType;

     struct RecordSetStatusType {
          CurrentSizeType currentSize;
          OperationalStateType operationalState;
```

```
          MaxSizeType maxSize;
          SizeThresholdType sizeThreshold;
          NameType filterName;
          FullActionType fullAction;
          AdministrativeStateType administrativeState;
          RecordKindType recordKind;
          UserLabelType recordSetUserLabel;
     };


     enum CreationModeType {
          operatorDefined,
          initialList,
          either
     };

// Local exceptions

     exception RecordSetExists {ManagedEntityIdType recordSetId;};
     exception LockedAlready {};
     exception UnknownOption {};
     exception NoSuchRecords {};
     exception TooManyRecords {};

// End local definitions



     interface RecordSetMgr : itut_x780::ManagedObject {


     // See 9.4.1.1 for the description of the behaviour of this operation

          ManagedEntityIdType createLog (
               in UserLabelType recordSetUserLabel,
               in AdministrativeStateType administrativeState,
               in NameType filterName,
               in FullActionType fullAction,
               in MaxSizeType maxSize,
               in SizeThresholdType sizeThreshold)
               raises (RecordSetExists,
                    DuplicateUserLabel,
                    AccessDenied);

     // See 9.4.1.2 for the description of the behaviour of this operation

          ManagedEntityIdType createArchive (
               in UserLabelType recordSetUserLabel,
               in AdministrativeStateType administrativeState,
               in RecordKindType recordKind,
               in MaxSizeType maxSize)
               raises (RecordSetExists,
                    DuplicateUserLabel,
                    AccessDenied);

     // See 9.4.1.3 for the description of the behaviour of this operation


          RecordSetStatusType getStatusAttributes (
               in ManagedEntityIdType recordSetId)
               raises (AccessDenied, UnknownRecordSet);

     // See 9.4.1.4 for the description of the behaviour of this operation
```

```
        void suspendArchive (
             in ManagedEntityIdType recordSetId)
             raises (AccessDenied, UnknownRecordSet);

    // See 9.4.1.5 for the description of the behaviour of this operation

        void resumeArchive (
             in ManagedEntityIdType recordSetId)
             raises (AccessDenied, UnknownRecordSet);

    // See 9.4.1.6 for the description of the behaviour of this operation

        void deleteArchive (
             in ManagedEntityIdType recordSetId )
             raises (UnknownRecordSet,
                  AccessDenied );

    // See 9.4.1.7 for the description of the behaviour of this operation

        void purgeArchive (
             in ManagedEntityIdType recordSetId )
             raises (UnknownRecordSet,
                  AccessDenied );


    // See 9.4.1.8 for the description of the behaviour of this operation

        RecordSeqType selectRecords (
             in FilterType SelectionFilter,
             in ManagedEntityIdType recordSetId)
             raises (UnknownRecordSet,
                  Timeout,
                  NoSuchRecords,
                  AccessDenied,
                  TooManyRecords);

    // See 9.4.1.9 for the description of the behaviour of this operation

        ManagedEntityIdSeqType recordSetListGet (
             in CreationModeType creationMode)
             raises (AccessDenied);


    // See 9.4.1.10 for the description of the behaviour of this operation

        void changeUserLabel(
             in ManagedEntityIdType recordSetId,
             in UserLabelType newUserLabel)
             raises (UnknownRecordSet,
                  AccessDenied,
                  DuplicateUserLabel);

    }; // interface RecordSetMgr

}; // module ControlArchive

}; // module q834_4

#endif
```

C.5 Q834SoftwareDownload.idl

```
#ifndef __Q834_4_SOFTWAREDOWNLOAD_DEFINED
#define __Q834_4_SOFTWAREDOWNLOAD_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module SoftwareDownload {

// begin definitions from other idl files

// From Q834Common
     typedef Q834Common::DCNAddressType DCNAddressType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::VersionType VersionType;
     typedef Q834Common::PlugInUnitType PlugInUnitType;
     typedef Q834Common::FilenameType  FilenameType;
     typedef Q834Common::ProceduralStatusSetType ProceduralStatusSetType;
     typedef Q834Common::StatusAttributeSeqType StatusAttributeSeqType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;
     typedef Q834Common::StatusValueType StatusValueType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::TrackingObjectIdType TrackingObjectIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define UnknownScheduler Q834Common::UnknownScheduler
#define UnknownNE Q834Common::UnknownNE
#define DeniedAccess Q834Common::DeniedAccess
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define Timeout Q834Common::Timeout
#define InvalidScheduler Q834Common::InvalidScheduler
#define InvalidStartTime Q834Common::InvalidStartTime

// End definitions from other idl files

// Local data types

     typedef TrackingObjectIdType SoftwareDownloadTrackingObjectIdType;

     typedef sequence<SoftwareDownloadTrackingObjectIdType>
SoftwareDownloadTrackingObjectIdSeqType;
     typedef sequence<FilenameType> FilenameSeqType;

     struct VersionsType {
          ManagedEntityIdType resourceId;
          VersionType softwarePrimary;
          VersionType softwareStandBy;
          VersionType hardware;
     };

     typedef sequence<VersionsType> VersionsSeqType;

     struct TargetType {
          ManagedEntityIdType containingSystem;
          ManagedEntityIdType containingNE; // empty sequence means everything
```

```
          string plugInUnitType; // empty string here implies to any suitable
plugInUnitType
          ManagedEntityIdType slot; // empty sequence here means any slot.
     }; //string is supplied by the supplier with Release Notes.

     struct DownloadStatusType
     {
             ManagedEntityIdType targetId;
             StatusValueType deliveryStatus;
             StatusValueType commitStatus;
             StatusValueType activationStatus;
             StatusValueType revertStatus;
     };

        typedef sequence<DownloadStatusType> DownloadStatusSeqType;

// Local exceptions


     exception InvalidExternalTime {};
     exception UnrecognisedTarget {};
     exception InsufficientMemory {};
     exception SoftwareLoadHWMismatch {};
     exception SourceUnreachable {};
     exception UnknownSoftwareLoad {};
     exception InstallationFailure {};
     exception UnknownSoftwareDownloadTrackingObject {};
     exception SoftwareNotYetInstalled {};
     exception ActivationFailure {};
     exception ActivationCompleted {};
     exception ActivityCompleted {};
     exception ActivityInProgress {};
     exception InvalidSoftwareTrackingObject {};
     exception SoftwareTrackingObjectInUse {};

// End local definitions

     valuetype DownloadMgrValueType: itut_x780::ManagedObjectValueType {

          public SoftwareDownloadTrackingObjectIdSeqType
ScheduledSoftwareDownloadTrackingObjectList; // GET
          public SoftwareDownloadTrackingObjectIdSeqType
OnDemandSoftwareDownloadTrackingObjectList; // GET

     };

     interface DownloadMgr : itut_x780::ManagedObject {

     // See 9.5.1.1 for the description of the behaviour of this operation

          SoftwareDownloadTrackingObjectIdType deliverDistSWGlobal (
               in FilenameSeqType softwareSet,
               in DCNAddressType softwareSourceAddr,
               in UserIdType userId,
               in PasswordType password,
               in ManagedEntityIdSeqType deliverDistTargets)
               raises (CommFailure,
                    UnrecognisedTarget,
                    InsufficientMemory,
                    SoftwareLoadHWMismatch,
                    SourceUnreachable,
                    UnknownSoftwareLoad,
                    Timeout,
                    AccessDenied,
```

```
                DeniedAccess);

// See 9.5.1.2 for the description of the behaviour of this operation

     SoftwareDownloadTrackingObjectIdType deliverDistSWSpecific (
          in FilenameSeqType softwareSet,
          in DCNAddressType softwareSourceAddr,
          in UserIdType userId,
          in PasswordType password,
          in TargetType deliverDistTarget)
          raises (CommFailure,
               UnrecognisedTarget,
               InsufficientMemory,
               SoftwareLoadHWMismatch,
               SourceUnreachable,
               UnknownSoftwareLoad,
               Timeout,
               AccessDenied,
               DeniedAccess);

// See 9.5.1.3 for the description of the behaviour of this operation

     void deleteSoftwareDownloadTrackingObject (
          in SoftwareDownloadTrackingObjectIdType id)
          raises (UnknownSoftwareDownloadTrackingObject, AccessDenied);

// See 9.5.1.4 for the description of the behaviour of this operation

     void commit (
          in SoftwareDownloadTrackingObjectIdType id,
          in TargetType commitTarget)
          raises (InstallationFailure,
               UnknownSoftwareDownloadTrackingObject,
               AccessDenied,
               UnrecognisedTarget);

// See 9.5.1.5 for the description of the behaviour of this operation

     void activate (
          in SoftwareDownloadTrackingObjectIdType id,
          in TargetType activateTarget)
          raises (UnknownSoftwareDownloadTrackingObject,
               SoftwareNotYetInstalled,
               ActivationFailure,
               AccessDenied,
               UnrecognisedTarget);

// See 9.5.1.6 for the description of the behaviour of this operation

     void revert (
          in SoftwareDownloadTrackingObjectIdType id,
          in TargetType revertTarget)
          raises (UnknownSoftwareDownloadTrackingObject,
               SoftwareNotYetInstalled,
               ActivationFailure,
               AccessDenied,
               UnrecognisedTarget,
               InvalidSoftwareTrackingObject);

// See 9.5.1.7 for the description of the behaviour of this operation

     DownloadStatusSeqType getStatus (
          in SoftwareDownloadTrackingObjectIdType id)
```

```
          raises (UnknownSoftwareDownloadTrackingObject,
               AccessDenied);

// See 9.5.1.8 for the description of the behaviour of this operation

     SoftwareDownloadTrackingObjectIdType scheduleDeliverDist (
          in FilenameSeqType softwareSet,
          in DCNAddressType softwareSourceAddr,
          in UserIdType userId,
          in PasswordType password,
          in ManagedEntityIdSeqType deliverDistTargets,
          in GeneralizedTimeType deliverDistStartTime)
          raises (SoftwareLoadHWMismatch,
               UnknownScheduler,
               AccessDenied,
               InvalidStartTime);

// See 9.5.1.9 for the description of the behaviour of this operation

     void scheduleCommit (
          in SoftwareDownloadTrackingObjectIdType
          deliverDistSoftwareDownloadTrackingObjectId,
          in GeneralizedTimeType commitStartTime)
          raises (UnknownSoftwareDownloadTrackingObject,
               UnknownScheduler,
               SoftwareNotYetInstalled,
               AccessDenied,
               InvalidStartTime);

// See 9.5.1.10 for the description of the behaviour of this operation

     void scheduleActivate (
          in SoftwareDownloadTrackingObjectIdType id,
          in UserLabelType activateSchedulerName,
          in GeneralizedTimeType activateStartTime)
          raises (UnknownSoftwareDownloadTrackingObject,
               InvalidStartTime,
               SoftwareNotYetInstalled,
               AccessDenied,
               InvalidScheduler );

// See 9.5.1.11 for the description of the behaviour of this operation

     void cancelScheduledSoftwareActivity (
          in SoftwareDownloadTrackingObjectIdType id)
          raises (UnknownSoftwareDownloadTrackingObject,
               ActivityCompleted,
               ActivityInProgress,
               AccessDenied);

// See 9.5.1.12 for the description of the behaviour of this operation

     SoftwareDownloadTrackingObjectIdSeqType
scheduledSoftwareDownloadTrackingObjectList ()
          raises (AccessDenied);

// See 9.5.1.13 for the description of the behaviour of this operation

     SoftwareDownloadTrackingObjectIdSeqType
onDemandSoftwareDownloadTrackingObjectList ()
          raises (AccessDenied);


}; // interface DownloadMgr
```

```
     interface VersionRepository : itut_x780::ManagedObject {

     // See 9.5.2.1 for the description of the behaviour of this operation

          VersionsSeqType retrieveVersions (
               in ManagedEntityIdType containingManagedEntityId)
               raises (CommFailure,
                    UnknownManagedEntity,
                    AccessDenied);

     // See 9.5.2.2 for the description of the behaviour of this operation

          boolean validateNEVersion (
               in ManagedEntityIdType managedEntityId,
               in VersionType proposedSoftware)
               raises (UnknownNE, AccessDenied);

     // See 9.5.2.3 for the description of the behaviour of this operation

          boolean validatePlugInVersion (
               in ManagedEntityIdType plugInUnitId,
               in VersionType proposedSoftware)
               raises (UnknownManagedEntity, AccessDenied);

}; // interface VersionRepository

}; // module SoftwareDownload

}; // module q834_4

#endif
```

C.6 Q834EventPublisher.idl

```
#ifndef __Q834_4_EVENTPUBLISHING_DEFINED
#define __Q834_4_EVENTPUBLISHING_DEFINED
#include "Q834Common.idl"
#pragma prefix "itu.Int"


module q834_4
{

module EventPublisher
{

// begin definitions from other idl files - filterable data value types


// From Q834Common

     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::NameType NameType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::VersionType VersionType;
     typedef Q834Common::ProceduralStatusSetType ProceduralStatusSetType;
     typedef Q834Common::PerceivedSeverityType PerceivedSeverityType;
     typedef Q834Common::OperationalStateType OperationalStateType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
```

```
     typedef Q834Common::ProbableCauseType ProbableCauseType; // Values defined
in Q834Common::ProbableCauseConst
     typedef Q834Common::NotificationIdentifierType NotificationIdentifierType;
     typedef Q834Common::NotificationIdentifierSeqType
NotificationIdentifierSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::EquipmentHolderFType EquipmentHolderFType;
     typedef Q834Common::PlugInUnitFType PlugInUnitFType;
     typedef Q834Common::UsageStateType UsageStateType;
     typedef Q834Common::ControlStatusSetType ControlStatusSetType;
     typedef Q834Common::SpecificProblemSetType SpecificProblemSetType;
     typedef Q834Common::BackedUpStatusType BackedUpStatusType;
     typedef Q834Common::AttributeChangeSetType AttributeChangeSetType;
     typedef Q834Common::SecurityAlarmCauseType SecurityAlarmCauseType;

// End definitions from other idl files

// Local data types




     typedef NotificationIdentifierSeqType CorrelatedNotificationType;

     typedef AttributeChangeSetType StateChangeDefinitionType;


     struct ThresholdInfoType {
          MonitoredParameterType monitoredParameter;
          unsigned long long observedValue;
          unsigned long long thresholdValueLow;
          unsigned long long thresholdValueHigh;
     };

     /*
     Values for monitoredParameter are provided in
     Q834Common::MonitoringParameter.
     Currently only counters are supported for performance or traffic monitoring
     in BPON technology, consequently observedValue, and thresholdValueHigh are
     restricted to non-negative integer values, and thresholdValueLow is 0.
     */

     typedef   unsigned shortEquipmentType;




     enum ServiceAffectingType {
          serviceAffecting,
          nonServiceAffecting,
          unableToDetermine
     };


//End local data type definitions

// Local exceptions

// End local exceptions


     interface AlarmEventSupplier : itut_x780::ManagedObject {
```

```
/* Structured event fixed header mappings:
domain_type is set to "telecommunications",
type_name is set to "Alarm", and
event_name is set to one of the following constant strings provided
below.
*/

const string communicationAlarm = "CommunicationAlarm";
const string equipmentAlarm = "EquipmentAlarm";
const string environmentalAlarm = "EnvironmentalAlarm";
const string processingErrorAlarm = "ProcessingErrorAlarm";
const string qualityOfServiceAlarm = "QualityOfServiceAlarm";

/*
Filterable data names for populating the filterable event body in the
structured event are listed in order below.
*/

const string alarmEmittingMEId = "AlarmEmittingMEId";
const string eventTime = "EventTime";
const string probableCause = "ProbableCause";
const string specificProblems = "SpecificProblems";
const string perceivedSeverity = "PerceivedSeverity";
const string backUpStatus = "BackUpStatus";
const string backUpManagedEntityId = "BackUpManagedEntityId";
const string thresholdInfo = "ThresholdInfo";
const string notificationIdentifier = "NotificationIdentifier";
const string correlatedNotifications = "CorrelatedNotifications";
const string stateChangeDefinition = "StateChangeDefinition";
const string monitoredAttributes = "MonitoredAttributes";
const string additionalText = "AdditionalText";
const string serviceAffectingInd = "ServiceAffectingInd";

/*
Remaining filterable data values.


/* Names for State and Status Variables for StateChangeDefinition
*/

const string administrativeState = "AdministrativeState";
const string operationalState = "OperationalState";
const string usageState = "UsageState";
const string availabilityStatus = "AvailabilityStatus";
const string proceduralStatus = "ProceduralStatus";
const string controlStatus = "ControlStatus";
const string alarmStatus = "AlarmStatus";


/*
Mapping to filterable data within the structured event is provided
below for communication alarm, equipment alarm, processing error
alarm, environmental alarm, and quality of service alarm.

{
{"AlarmEmittingMEId", any (ManagedEntityIdType)},
{"EventTime", any (GeneralizedTimeType)},
{"ProbableCause", any (ProbableCauseType)},
{"SpecificProblems", any (SpecificProblemSetType)},
{"PerceivedSeverity", any (PerceivedSeverityType)},
{"ServiceAffectingInd", any (ServiceAffectingType)},
{"BackUpStatus", any (BackedUpStatusType)},
{"BackUpManagedEntityId", any (ManagedEntityIdType)},
{"ThresholdInfo", any (ThresholdInfoType)},
```

```
     {"NotificationIdentifier", any (NotificationIdentifierType)},
     {"CorrelatedNotifications", any (CorrelatedNotificationType)},
     {"StateChangeDefinition", any (StateChangeDefinitionType)},
     {"AdditionalText", any (string)}
     }

     */

}; // interface AlarmEventSupplier


interface SecurityEventSupplier : itut_x780::ManagedObject {

     /*  Structured event fixed header mappings:
     domain_type is set to "telecommunications",
     type_name is set to "SecurityEvent", and
     event_name is set to one of the following constant strings
     provided below.
     */

     const string integrityViolation = "IntegrityViolation";
     const string operationalViolation = "OperationalViolation";
     const string physicalViolation = "PhysicalViolation";
     const string securityEventViolation = "SecurityEventViolation";
     const string timeDomainViolation = "TimeDomainViolation";



     const string eventEmittingMEId = "EventEmittingMEId";
     const string eventTime = "EventTime";
     const string securityAlarmCause = "SecurityAlarmCause";
     const string securityAlarmDetector = "SecurityAlarmDetector";
     const string serviceUser = "ServiceUser";
     const string serviceProvider = "ServiceProvider";
     const string securityEventNotificationIdentifier =
          "NotificationIdentifier";
     const string correlatedNotifications = "CorrelatedNotifications";
     const string additionalText = "AdditionalText";
     /*
     Mapping to filterable data within the structured event is provided
     below for Integrity Violations, Operational Violations, Physical
     Violations,Security Event Violations, Time Domain Violations.


     {
     {"EventEmittingMEId", any (ManagedEntityIdType)},
     {"EventTime", any (GeneralizedTimeType)},
     {"SecurityAlarmCause", any (SecurityAlarmCauseType)},
     {"SecurityAlarmDetector", any (ManagedEntityIdType)},
     {"ServiceUser", any (ManagedEntityIdType)},
     {"ServiceProvider", any (ManagedEntityIdType)},
     {"NotificationIdentifier", any (NotificationIdentifierType)},
     {"CorrelatedNotifications", any (CorrelatedNotificationType)},
     {"AdditionalText", any (string)}
     }

     */

}; // interface SecurityEventSupplier


interface DiscoveryEventSupplier : itut_x780::ManagedObject {

     /*  Structured event fixed header mappings:
```

```
        domain_type is set to "telecommunications",
        type_name is set to "DiscoveryEvent", and
        event_name is set to one of the following constant strings
        provided below. Only installed equipment changes are declared
        through this interface.
        */

        const string managedEntityCreation = "ManagedEntityCreation";
        const string managedEntityDeletion = "ManagedEntityDeletion";


        const string managedEntityType = "ManagedEntityType";
        const string managedEntityAttributeValues =
        "ManagedEntityAttributeValues";
        const string discoveryNotificationIdentifier =
        "NotificationIdentifier";
        const string correlatedNotifications = "CorrelatedNotifications";
        const string additionalText = "AdditionalText";"


        // The following items are equipment types that are discovered by the
        // Supplier Management System and automatically revealed to the OMS.
        // The data structure passed on the notification is defined in
        // Q834Common.idl and the mapping below refers to it as MEstruct. More
        // specifically, here is the list of data structures currently
        // supported: OLTType, ONUType,ONTType, NTType, EquipmentHolderType,
        and PlugInUnitFType.

        const unsigned short OLT_NE = 1;
        const unsigned short ONT_NE = 2;
        const unsigned short ONU_NE = 3;
        const unsigned short NT_NE = 4;
        const unsigned short EQUIPMENT_HOLDER_F = 5;
        const unsigned short PLUG_IN_UNIT_F = 6;


        /*
        Mapping to filterable data within the structured event is provided
        below for a discovery event that involves the creation of a managed
        entity.


        {
        {"ManagedEntityType", any (EquipmentType)},
        {"EventTime", any (GeneralizedTimeType)},
        {"ManagedEntityAttributeValues", any (MEstruct)},
        {"NotificationIdentifier", any (NotificationIdentifierType)},
        {"CorrelatedNotifications", any (CorrelatedNotificationType)},
        {"AdditionalText", any (string)}
        }

        */

    }; // interface DiscoveryEventSupplier

}; // module EventPublisher

}; // module q834_4

#endif
```

C.7 Q834MIBTransfer.idl

```
#ifndef __Q834_4_MIBTRANSFER_DEFINED
#define __Q834_4_MIBTRANSFER_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module MIBTransfer {
// Begin definitions from other idl files


// From Q834Common

     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::DCNAddressType DCNAddressType;
     typedef Q834Common::FilenameType FilenameType;
     typedef Q834Common::StatusAttributeSeqType StatusAttributeSeqType;
     typedef Q834Common::TransferTrackingObjectIdType
     TransferTrackingObjectIdType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define EquipmentFailure Q834Common::EquipmentFailure
#define UnknownNE Q834Common::UnknownNE
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidScheduler Q834Common::InvalidScheduler
#define UnknownDestinationServer Q834Common::UnknownDestinationServer
#define FileExists Q834Common::FileExists
#define CannotCreateFile Q834Common::CannotCreateFile
#define DeniedAccess Q834Common::DeniedAccess

// End definitions from other idl files

// Local data types

     typedef sequence<TransferTrackingObjectIdType>
TransferTrackingObjectIdSeqType;

// Local exceptions

     exception UnknownBackupProcess {};
     exception UnknownSourceServer {};
     exception UnknownRestoreProcess {};
     exception SoftwareLoadHardwareMismatch {};


// End local definitions

     valuetype MIBMoverValueType: itut_x780::ManagedObjectValueType {

          public TransferTrackingObjectIdSeqType transferTrackingObjectIdList;
// GET

     };

     interface MIBMover : itut_x780::ManagedObject {
```

```
// See 9.7.1.1 for the description of the behaviour of this operation

     TransferTrackingObjectIdType startBackup (
          in ManagedEntityIdType nEManagedEntityId, // OLT
          in DCNAddressType destinationServerAddr,
          in UserIdType userId,
          in PasswordType password,
          in FilenameType destinationFile,
          in boolean overwriteExistingFile)
          raises (AccessDenied,
               UnknownNE,
               UnknownDestinationServer,
               CommFailure,
               EquipmentFailure,
               DeniedAccess);

// See 9.7.1.2 for the description of the behaviour of this operation

     StatusAttributeSeqType getBackupStatus (
          in TransferTrackingObjectIdType id)
          raises (AccessDenied,
               UnknownBackupProcess);


// See 9.7.1.3 for the description of the behaviour of this operation

     TransferTrackingObjectIdType scheduleBackup (
          in ManagedEntityIdType nEManagedEntityId, // OLT
          in UserIdType userId,
          in PasswordType password,
          in UserLabelType schedulerName,
          in DCNAddressType destinationServerAddr,
          in FilenameType destinationFile,
          in boolean overwriteExistingFile)
          raises (AccessDenied,
               UnknownNE,
               UnknownScheduler,
               UnknownDestinationServer,
               InvalidScheduler);

// See 9.7.1.4 for the description of the behaviour of this operation

         void modifyBackupSchedule(
          in TransferTrackingObjectIdType id,
          in UserLabelType newSchedulerName)
          raises (AccessDenied,
               UnknownBackupProcess,
               UnknownScheduler,
               InvalidScheduler);

// See 9.7.1.5 for the description of the behaviour of this operation

     void cancelScheduledBackup (
          in TransferTrackingObjectIdType id)
          raises (AccessDenied,
               UnknownBackupProcess);

// See 9.7.1.6 for the description of the behaviour of this operation

     void abortBackup (
          in TransferTrackingObjectIdType id)
          raises (AccessDenied,
               UnknownBackupProcess,
               CommFailure,
```

```
                    EquipmentFailure);

     // See 9.7.1.7 for the description of the behaviour of this operation

          TransferTrackingObjectIdType startRestore (
               in ManagedEntityIdType nEManagedEntityId, // OLT
               in DCNAddressType sourceServerAddr,
               in UserIdType userId,
               in PasswordType password,
               in FilenameType sourceFile)
               raises (AccessDenied,
                    UnknownNE,
                    UnknownSourceServer,
                    CommFailure,
                    EquipmentFailure,
                    DeniedAccess,
                    SoftwareLoadHardwareMismatch );

     // See 9.7.1.8 for the description of the behaviour of this operation

          StatusAttributeSeqType getRestoreStatus (
               in TransferTrackingObjectIdType id)
               raises (AccessDenied,
                    UnknownRestoreProcess);

     // See 9.7.1.9 for the description of the behaviour of this operation

          TransferTrackingObjectIdSeqType transferTrackingObjectIdListGet ()
               raises (AccessDenied);

     }; // interface MIBMover

}; // module MIBTransfer

}; // module q834_4

#endif
```

C.8 Q834PerformanceManager.idl

```
#ifndef __Q834_4_PERFORMANCEMANAGER_DEFINED
#define __Q834_4_PERFORMANCEMANAGER_DEFINED


#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module PerformanceManager {

     // begin definitions from other idl files

     // From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::RecordSeqType RecordSeqType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::NameSeqType NameSeqType;
```

```
    typedef Q834Common::MonitoringKindType MonitoringKindType;
    typedef Q834Common::RecordKindType RecordKindType;


    #define AccessDenied Q834Common::AccessDenied
    #define UnknownNE Q834Common::UnknownNE
    #define UnknownManagedEntity Q834Common::UnknownManagedEntity
    #define UnknownProfiles Q834Common::UnknownProfiles
    #define UnknownServiceInstance Q834Common::UnknownServiceInstance
    #define UnknownScheduler Q834Common::UnknownScheduler
    #define CommFailure Q834Common::CommFailure
    #define InvalidScheduler Q834Common::InvalidScheduler
    #define EquipmentFailure Q834Common::EquipmentFailure
    #define ProfileSuspended Q834Common::ProfileSuspended

    // End definitions from other idl files

    // Local data types


    struct  MonitoringPointKindAndThresholdsType {
         MonitoringKindType monitoringType;
         NameType  thresholdDataProfileName;
    };
    /*
    NOTE – ITU-T Recs Q.834.1 and Q.834.2 describe the relationships between
    ThresholdData and specific monitoring point types.
    */

    typedef sequence<MonitoringPointKindAndThresholdsType> ThresholdsSeqType;

    struct MonitoringPointAndThresholdsType {
         ManagedEntityIdType monitoringPoint;
         NameType  thresholdDataProfileName;
    };

    typedef sequence<MonitoringPointAndThresholdsType>
MonitoringPointThresholdsSeqType;

    typedef sequence<MonitoredParameterType> MonitoredParameterSeqType;

    typedef sequence<RecordSeqType> RecordsSeqType; // Implicitly grouped by
record type.

    typedef RecordKindType  HistoryDataType; // Values defined in
Q834Common::RecordSetType

    typedef ManagedEntityIdSeqType MonitoringPointSeqType;

    struct SWPValueType {
         ManagedEntityIdType monitoringPoint;
         short totConsecutiveIntvls;
         short persistenceMinimum;
    };

    typedef sequence< SWPValueType > SWPValueSeqType;


    struct ParameterSettingType {
         MonitoredParameterType monitoredParameter;
         short totConsecutiveIntvls;
         short persistenceMinimum;
    };
```

```
typedef sequence<ParameterSettingType> ParameterSettingSeqType;

// Local exceptions

exception UnknownParameters {
     MonitoredParameterSeqType monitoredParameterList;
};
exception IntervalCountTooLarge {};
exception UnknownMonitoringPointTypes {
     MonitoringPointSeqType monitoringPointKindList;
};
exception InvalidAssociation {};
exception CollectionPeriodPast {};
exception CollectionLimitation {};
exception UnknownHistoryDataType {};

// End local definitions


interface ImpairmentPersistence : itut_x780::ManagedObject {

     // See 9.8.1.1 for the description of the behaviour of this operation

     void setSlidingWindowParameters (
          in ManagedEntityIdType nEManagedEntityId,
          in MonitoredParameterSeqType monitoredParameterList,
          in short totConsecutiveIntvls,
          in short persistenceMinimum,
          in boolean sysScopeInd)
          raises (UnknownNE,
               UnknownParameters,
               IntervalCountTooLarge,
               AccessDenied,
               CommFailure);

     // See 9.8.1.2 for the description of the behaviour of this operation

     void setSpecificSlidingWindowParameters (
          in ManagedEntityIdType nEManagedEntityId,
          in ManagedEntityIdType monitoringPoint,
          in MonitoredParameterSeqType monitoredParameterList,
          in short totConsecutiveIntvls,
          in short persistenceMinimum,
          in boolean allowGlobalOverwrite)
          raises (UnknownNE,
               UnknownParameters,
               IntervalCountTooLarge,
               AccessDenied,
               UnknownManagedEntity,
               CommFailure,
               EquipmentFailure);

     // See 9.8.1.3 for the description of the behaviour of this operation

     SWPValueSeqType getSpecificSlidingWindowParameters (
          in ManagedEntityIdType nEManagedEntityId,
          in MonitoredParameterType monitoredParameter)
          raises (UnknownNE,
               UnknownParameters,
               CommFailure);

     // See 9.8.1.4 for the description of the behaviour of this operation

     void setThreshold (
```

```
            in ManagedEntityIdType nEManagedEntityId,
            in ManagedEntityIdType monitoringPoint,
            in NameType thresholdDataProfileName)
            raises (UnknownNE,
                 AccessDenied,
                 UnknownManagedEntity,
                 UnknownProfiles,
                 InvalidAssociation,
                 CommFailure,
                 ProfileSuspended);

       // See 9.8.1.5 for the description of the behaviour of this operation

       void setThresholds (
            in ManagedEntityIdType nEManagedEntityId,
            in boolean sysScopeInd,
            in ThresholdsSeqType thresholdsList)
            raises (UnknownNE,
                 UnknownProfiles,
                 AccessDenied,
                 UnknownMonitoringPointTypes,
                 InvalidAssociation,
                 CommFailure,
                 ProfileSuspended);

       // See 9.8.1.6 for the description of the behaviour of this operation

       MonitoringPointThresholdsSeqType getThresholdValues (
            in ManagedEntityIdType nEManagedEntityId,
            in MonitoringKindType monitoringType)
            raises (UnknownNE,
                 UnknownMonitoringPointTypes,
                 CommFailure);

       // See 9.8.1.7 for the description of the behaviour of this operation

       ThresholdsSeqType getSystemThresholdsSetting(
            in ManagedEntityIdType nEManagedEntityId)
            raises (AccessDenied, UnknownManagedEntity);

       // See 9.8.1.8 for the description of the behaviour of this operation

       ParameterSettingSeqType getSystemSWSettings(
            in ManagedEntityIdType nEManagedEntityId)
            raises (AccessDenied, UnknownManagedEntity);

  }; // interface ImpairmentPersistence


  interface ReportController : itut_x780::ManagedObject {

       // See 9.8.2.1 for the description of the behaviour of this operation

       void addCustomerMonitoringReporting  (
            in ManagedEntityIdType nEManagedEntityId,
            in ServiceInstanceIdType serviceInstanceId,
            in ManagedEntityIdType monitoringPoint,
            in GeneralizedTimeType stopTime,
            in HistoryDataType historyData,
            in short granularityPeriod) //in minutes
            raises (UnknownServiceInstance,
                 AccessDenied,
                 UnknownNE,
                 UnknownManagedEntity,
```

```
          CollectionPeriodPast,
          CollectionLimitation,
          InvalidAssociation,
          UnknownHistoryDataType,
          CommFailure);

// See 9.8.2.2 for the description of the behaviour of this operation

void removeCustomerMonitoringReporting  (
     in ServiceInstanceIdType serviceInstanceId)
     raises (UnknownServiceInstance,
          AccessDenied,
          CollectionPeriodPast,
          CommFailure);

// See 9.8.2.3 for the description of the behaviour of this operation

RecordsSeqType selectByServiceInstance (
     in ServiceInstanceIdType serviceInstanceId,
     in GeneralizedTimeType intervalStartTime,
     in GeneralizedTimeType intervalEndTime)
     raises (UnknownServiceInstance,
          AccessDenied );

// See 9.8.2.4 for the description of the behaviour of this operation

MonitoringPointSeqType displayActiveReporting (
     in ServiceInstanceIdType serviceInstanceId)
     raises (UnknownServiceInstance,
          AccessDenied);

// See 9.8.2.5 for the description of the behaviour of this operation

void addNewMonitoringReporting (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in GeneralizedTimeType stopTime,
     in HistoryDataType historyData,
     in short granularityPeriod) //in minutes
     raises (AccessDenied,
          UnknownNE,
          UnknownManagedEntity,
          CollectionPeriodPast,
          CollectionLimitation,
          InvalidAssociation,
          UnknownHistoryDataType,
          CommFailure);

// See 9.8.2.6 for the description of the behaviour of this operation

RecordsSeqType selectByMonitoringPoint (
     in ManagedEntityIdType monitoringPoint,
     in GeneralizedTimeType intervalStartTime,
     in GeneralizedTimeType intervalEndTime)
     raises (UnknownManagedEntity,
          AccessDenied);


// See 9.8.2.7 for the description of the behaviour of this operation

void createReportingSchedule (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in HistoryDataType historyData,
```

```
                in ServiceInstanceIdType serviceInstance,
                in short granularityPeriod, //in minutes
                in UserLabelType schedulerName)
                raises (AccessDenied,
                     UnknownNE,
                     UnknownManagedEntity,
                     CollectionLimitation,
                     UnknownScheduler,
                     InvalidAssociation,
                     UnknownHistoryDataType,
                     InvalidScheduler);

           // See 9.8.2.8 for the description of the behaviour of this operation

           void modifyReportingSchedule (
                in ManagedEntityIdType nEManagedEntityId,
                in ManagedEntityIdType monitoringPoint,
                in HistoryDataType historyData,
                in UserLabelType newSchedulerName)
                raises (AccessDenied,
                     UnknownNE,
                     UnknownManagedEntity,
                     CollectionLimitation,
                     UnknownScheduler,
                     InvalidAssociation,
                     UnknownHistoryDataType,
                     InvalidScheduler);

           // See 9.8.2.9 for the description of the behaviour of this operation

           void cancelReportingSchedule (
                in ManagedEntityIdType nEManagedEntityId,
                in ManagedEntityIdType monitoringPoint,
                in HistoryDataType historyData,
                in UserLabelType schedulerName)
                raises (AccessDenied,
                     UnknownNE,
                     UnknownManagedEntity,
                     UnknownScheduler,
                     InvalidAssociation,
                     UnknownHistoryDataType);

     }; // interface ReportController

}; // module PerformanceManager

}; // module q834_4

#endif
```

C.9 Q834ProfileManager.idl

```
#ifndef __Q834_4_PROFILEMANAGER_DEFINED
#define __Q834_4_PROFILEMANAGER_DEFINED
#include "Q834Common.idl"
#pragma prefix "itu.Int"


module  q834_4
{

module ProfileManager
{
```

```
     // begin definitions from other idl files - filterable data value types
     typedef Q834Common::NameType NameType;
     typedef Q834Common::NotificationIdentifierType NotificationIdentifierType;
     typedef Q834Common::PerceivedSeverityType PerceivedSeverityType;
     typedef Q834Common::ProbableCauseType ProbableCauseType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::ServiceCategoryType ServiceCategoryType;
     typedef Q834Common::ConformanceDefType ConformanceDefType;
     typedef Q834Common::ATMOverbookingFactorType ATMOverbookingFactorType;
     typedef Q834Common::MonitoringKindType MonitoringKindType;

#define UnknownProfiles Q834Common::UnknownProfiles
#define AccessDenied Q834Common::AccessDenied

     // End definitions from other idl files

     // Local data types

     typedef unsigned short ProfileKindType;

     struct ProfileInfoType {
          ProfileKindType profileKind;
          any  attributeValuestruct;
     };

     /*
     Next, structs (and sequence of structs) are defined having the attribute
     values for the newly created profile objects. Any one of these structs is
     subsequently identified as ProfileStruct in the Profile Event Consumer
     interface specification.
     */

     enum SubType  {
          NULL,
          VoicebandBasedOn64kbps,
          SynchronousCircuitEmulation,
          AsynchronousCircuitEmulation,
          HighQualityAudio,
          Video
     };

     enum CBRRateType {
          br44736kbps,
          br1544kbps,
          br2048kbps,
          brE3kbps,
          br64kbps,
          br2x64kbps,
          br3x64kbps,
          br4x64kbps,
          br5x64kbps,
          br6x64kbps,
          br7x64kbps,
          br8x64kbps,
          br9x64kbps,
          br10x64kbps,
          br11x64kbps,
          br12x64kbps,
          br13x64kbps,
          br14x64kbps,
          br15x64kbps,
          br16x64kbps,
          br17x64kbps,
```

```
     br18x64kbps,
     br19x64kbps,
     br20x64kbps,
     br21x64kbps,
     br22x64kbps,
     br23x64kbps,
     br24x64kbps,
     br25x64kbps,
     br26x64kbps,
     br27x64kbps,
     br28x64kbps,
     br29x64kbps,
     br30x64kbps,
     br31x64kbps
};

enum ClockRecoveryType {
     PhysInterface,
     SRTS,
     AdaptiveClock,
     LocalOsc
};

enum ForwardErrorCorrectionType {
     NoFEC,
     FECforLossSensitiveSigTransport,
     FECforDelaySensSigTransport
};

typedef boolean StructuredDataTransferType; //TRUE = STD has been chosen
typedef long long PartiallyFilledCellsType;
typedef long long CellLossIntegrationPeriodType;

struct AAL1ProfileType {
     SubType aAL1subtype;
     CBRRateType cBRRate;
     ClockRecoveryType clockRecovery;
     ForwardErrorCorrectionType forwardErrorCorrection;
     StructuredDataTransferType structuredDataTransfer;
     PartiallyFilledCellsType partiallyFilledCells;
}; // ProfileStruct for profile type = 1


typedef stringDefaultSSCSParameterProfile1PtrType;

typedef stringDefaultSSCSParameterProfile2PtrType;

struct    AAL2ProfileType {
     DefaultSSCSParameterProfile1PtrType defaultSSCSParameterProfile1Ptr;
     DefaultSSCSParameterProfile2PtrType defaultSSCSParameterProfile2Ptr;
}; // ProfileStruct for profile type = 2


struct    MaxCPCSSDUSizeType {
     long long forwardDirectionCPCS_SDU;
     long long backwardsDirectionCPCS_SDU;
};

enum AALModeType {
     MessageAssured,
     MessageUnassured,
     StreamingAssured,
     StreamingUnassured
};
```

```
     enum SSCSType {
          NoSSC,
          DataSSCSonSSCOPAssured,
          DataSSCSonSSCOPNotAssured,
          FrameRelaySSCS
     };

     struct AAL5ProfileType {
          // ManagedEntityIdType mEId; ??
          MaxCPCSSDUSizeType maxCPCSSDUSize;
          AALModeType aALMode;
          SSCSType sSCS;
     }; // ProfileStruct for profile type = 4


     enum AppIdType {
          LES_CAS_noELCP,
          LES_PSTN_noELCP,
          LES_PSTN_ELCP,
          LES_DSS1forBRI_noELCP,
          LES_DSS1forBRI_ELCP,
          LES_CASforPOTS_DSS1forBRI_noELCP,
          LES_PSTNforPOTS_DSS1forBRI_noELCP,
          LES_PSTNforPOTS_DSS1forBRI_ELCP,
          LES_otherCCS,
          UnspecifiedLES
     };

     typedef long long MaxNumChannelsType;
     typedef long long MinimumChanIdType;
     typedef long long MaximumChanIdType;
     typedef long long MaxCPS_SDULengthType;
     typedef long long TimerCULengthType;

     struct AAL2PVCProfileType {
          MaxNumChannelsType maxNumChannels;
          MinimumChanIdType minimumChanId;
          MaximumChanIdType maximumChanId;
          MaxCPS_SDULengthType maxCPS_SDULength;
          TimerCULengthType timerCULength;
     }; // ProfileStruct for profile type = 3

     struct AlarmSeverityAssignProfileType {
          string eventName; //As defined in AlarmEventSupplier
          ProbableCauseType probableCauseValue; //Ditto as above
          PerceivedSeverityType serviceAffectingSeverity;
          PerceivedSeverityType nonServiceAffectingSeverity;
     }; // ProfileStruct for profile type = 5

     typedef sequence<AlarmSeverityAssignProfileType>
AlarmSeverityAssignProfileSeqType;
     typedef long long LocalMaxNumVPCSupportedType;
     typedef long long LocalMaxNumVCCSupportedType;
     typedef short LocalMaxNumVPIBitsType;
     typedef short LocalMaxNumVCIBitsType;
     typedef long long TotalEgressBandwidthType;
     typedef long long TotalIngressBandwidthType;
     typedef boolean UPCNPCIndicatorType; //TRUE = policing is on

     struct ATMNetworkAccessProfileType {
          LocalMaxNumVPCSupportedType localMaxNumVPCSupported;
          LocalMaxNumVCCSupportedType localMaxNumVCCSupported;
          LocalMaxNumVPIBitsType localMaxNumVPIBits;
```

```
     LocalMaxNumVCIBitsType localMaxNumVCIBits;
     TotalEgressBandwidthType totalEgressBandwidth;
     TotalIngressBandwidthType totalIngressBandwidth;
     UPCNPCIndicatorType uPCNPCIndicator;
}; // ProfileStruct for profile type = 6


typedef short LANType;
typedef short EncapsulationProtocolType;
typedef short PIDType;

struct BridgedLANServiceProfileValues {
     LANType lAN;
     EncapsulationProtocolType encapsulationProtocol;
     PIDType pID;
}; // ProfileStruct for profile type = 7


typedef long long BufferedCDVToleranceType;

enum CASType {
     basic,
     e1Cas,
     SfCas,
     ds1EsfCas,
     j2Cas
};

typedef long long CableLengthType;

struct CESServiceProfileValues {
     BufferedCDVToleranceType bufferedCDVTolerance;
     CASType cAS;
}; // ProfileStruct for profile type = 8


enum DS1FramingType {
     SuperFrame,
     ExtendedSuperFrame,
     Unframed
};

enum DS1EncodingType {
     AMI,
     B8ZS
};

enum LoopbackCodeType {
     SmartJack,
     SmartJack_ONTInband,
     COInband
};

typedef boolean SupplyPowerIndType;

struct DS1ProfileValues {
     DS1FramingType dS1Framing;
     DS1EncodingType dS1Encoding;
     LoopbackCodeType loopbackCode;
     SupplyPowerIndType supplyPowerInd;
     CableLengthType cableLength;
}; // ProfileStruct for profile type = 9
```

```
enum DS3ApplicationType {
     CBitParity,
     M23
};

enum DS3EncodingType {
     B3ZS,
     CCHAN
};

struct DS3ProfileType {
     DS3ApplicationType dS3Application;
     DS3EncodingType dS3Encoding;
     CableLengthType cableLength;
}; // ProfileStruct for profile type = 10

typedef boolean DuplexType; //TRUE = full, FALSE = half
typedef boolean AutoDetectionIndType;

enum DataRateType {
     TenBT,
     HundredBT,
     ThousandBT,
     otherrate
};

typedef long long MaxFrameSizeType; //Fixed for Ethernet

typedef boolean DTEDCEType; // TRUE = DTE setting; FALSE = DCE setting.

struct EthernetProfileValues {
     DuplexType duplex;
     AutoDetectionIndType autoDetectionInd;
     DataRateType dataRate;
     MaxFrameSizeType maxFrameSize;
     DTEDCEType dTEDCE;
}; // ProfileStruct for profile type = 11


enum T303Type {
     m700,
     m1200,
     m1700,
     m2200,
     m2700,
     m3200,
     m3700,
     m4200,
     m4700
};

enum T396Type {
     ms700,
     ms1700,
     ms2700,
     ms3700,
     ms4700,
     ms5700,
     ms6700,
     ms7700,
     ms8700,
     ms9700,
     ms10700,
     ms11700,
```

```
     ms12700,
     ms13700,
     ms14700
};

struct IDLCCallProcessingProfileType {
     T303Type t303;
     T396Type t396;
}; // ProfileStruct for profile type = 12

typedef boolean ELCPIndType;

enum POTSSignallingType {
     PSTN,
     ChAS,
     CCS,
     UNKNOWN
};

enum BRISignallingType {
     DSS1,
     OtherCCS
};

typedef long long MaxNumCIDsType;
typedef long long MaxPacketLengthType;
struct ChannelWithSSCSPtrType {
     long channelIndex;
     boolean ptr1orPtr2;
};

typedef sequence<ChannelWithSSCSPtrType> ChanAndSSCSParaPtrSeqType;

struct LESProfileType {
     ELCPIndType eLCPInd;
     POTSSignallingType pOTSSignalling;
     BRISignallingType bRISignalling;
     MaxNumCIDsType maxNumCIDs;
     MaxPacketLengthType maxPacketLength;
     ChanAndSSCSParaPtrSeqType chanAndSSCSParaPtrSeq;
}; // ProfileStruct for profile type = 13


typedef boolean SpanningTreeIndType;
typedef short BridgePriorityType;
typedef short MaxAgeType;
typedef short HelloTimeType;
typedef short ForwardDelayType;

struct MACBridgeServiceProfileType {
     SpanningTreeIndType spanningTreeInd;
     BridgePriorityType bridgePriority;
     MaxAgeType maxAge;
     HelloTimeType helloTime;
     ForwardDelayType forwardDelay;
}; // ProfileStruct for profile type = 14

typedef long long SegmentLengthType;
typedef short RASTimerType;
typedef long longMaxSSSARSDULengthType;
typedef boolean SSTEDIndType;
typedef boolean SSADTIndType;

struct SSCSParameterProfile1Type {
```

```
     SegmentLengthType segmentLength;
     RASTimerType rASTimer;
     MaxSSSARSDULengthType maxSSSARSDULength;
     SSTEDIndType sSTEDInd;
     SSADTIndType sSADTInd;
}; // ProfileStruct for profile type = 15

enum ServiceCatType {
     Audio,
     Multirate,
     UNKNCategory
};

enum EncSrcType {
     ITUT,
     ATMForum,
     Proprietary
};

typedef short EncProfileIndexType;
typedef boolean AudioServIndType;
typedef short PCMEncType;
typedef boolean CMDataIndType;
typedef short CMMultiplierNumType;
typedef boolean FMDataIndType;
typedef long long FMMaxFrameLenType;
typedef boolean CASIndType;
typedef boolean DTMFIndType;
typedef boolean MFR1IndType;
typedef boolean MFR2IndType;
typedef boolean RateControlIndType;
typedef boolean SynchChangeIndType;
typedef boolean FaxDemodIndType;

struct SSCSParameterProfile2Type {
     ServiceCatType serviceCat;
     EncSrcType encSrc;
     EncProfileIndexType encProfileIndex;
     AudioServIndType audioServInd;
     PCMEncType pCMEnc;
     CMDataIndType cMDataInd;
     CMMultiplierNumType cMMultiplierNum;
     FMDataIndType fMDataInd;
     FMMaxFrameLenType fMMaxFrameLen;
     CASIndType cASInd;
     DTMFIndType dTMFInd;
     MFR1IndType mFR1Ind;
     MFR2IndType mFR2Ind;
     RateControlIndType rateControlInd;
     SynchChangeIndType synchChangeInd;
     FaxDemodIndType faxDemodInd;
}; // ProfileStruct for profile type = 16


typedef long long PCRIngressType;
typedef long long PCREgressType;
typedef long long CDVTPCRIngressType;
typedef long long CDVTPCREgressType;
typedef long long CDVTSCRIngressType;
typedef long long CDVTSCREgressType;
typedef long long SCRIngressType;
typedef long long SCREgressType;
typedef long long MaxBSIngressType;
typedef long long MaxBSEgressType;
```

```
typedef long long MFSIngressType;
typedef long long MFSEgressType;

struct TrafficDescriptorProfileType {
     ServiceCategoryType serviceCategory;
     ConformanceDefType conformanceDef;
     PCRIngressTypepCRIngress;
     PCREgressType pCREgress;
     CDVTPCRIngressType cDVTPCRIngress;
     CDVTPCREgressType cDVTPCREgress;
     CDVTSCRIngressType cDVTSCRIngress;
     CDVTSCREgressType cDVTSCREgress;
     SCRIngressType sCRIngress;
     SCREgressType sCREgress;
     MaxBSIngressType maxBSIngress;
     MaxBSEgressType maxBSEgress;
     MFSIngressType mFSIngress;
     MFSEgressType mFSEgress;
}; // ProfileStruct for profile type = 17


typedef string LoopbackLocCodeType;

struct UNIProfileType {
     LocalMaxNumVPCSupportedType localMaxNumVPCSupported;
     LocalMaxNumVCCSupportedType localMaxNumVCCSupported;
     LocalMaxNumVPIBitsType localMaxNumVPIBits;
     LocalMaxNumVCIBitsType localMaxNumVCIBits;
     LoopbackLocCodeType loopbackLocCode;
}; // ProfileStruct for profile type = 18


enum AnnouncementType {
     Silence,
     ReorderTone,
     FastBusy,
     VoiceAnnouncement,
     OtherAnnouncementType,
     UNKNAnnouncementType
};

enum TimingReferenceType {
     NetworkTimingReference,
     AdaptiveVoice,
     FreeRun,
     OtherTimingReference
};

typedef boolean EchoCancellationIndType;

struct VoiceServiceProfileAAL1Type {
     AnnouncementType announcement;
     TimingReferenceType timingReference;
     EchoCancellationIndType echoCancellationInd;
}; // ProfileStruct for profile type = 19


typedef long long JitterTargetType;
typedef long long JitterBufferMaxType;

struct VoiceServiceProfileAAL2Type {
     AnnouncementType announcement;
     TimingReferenceType timingReference;
     JitterTargetType jitterTarget;
```

```
        JitterBufferMaxType jitterBufferMax;
        EchoCancellationIndType echoCancellationInd;
    }; // ProfileStruct for profile type = 20

    struct ThresholdDataComponentType {
        MonitoredParameterType monitoredParameter; // Values defined in
        Q834Common::MonitoringParameter unsigned long long thresholdValue;
    };

    typedef sequence<ThresholdDataComponentType> ThresholdDataSeqType;

    struct ThresholdDataProfileType {
        MonitoringKindType monitoringType;
        ThresholdDataSeqType thresholdValues;
    }; // ProfileStruct for profile type = 21

    typedef sequence<ATMOverbookingFactorType> ATMOverbookingProfileType;
// ProfileStruct for profile type = 22

    // Local exceptions
    exception ProfileInUse {};
    exception DuplicateProfileName {};

    // End local definitions


    interface ProfileConsumer : itut_x780::ManagedObject {
        /*
        Structured event fixed header mappings:
        domain_type is set to "telecommunications",
        type_name is set to "ProfileEvent", and
        event_name is set to the constant string
        provided below. Only new profiles are announced
        through this interface.
        */

        const string profileCreation = "ProfileCreation";

        /*
        Identification of remaining items in filterable data of structured
        event.
        */

        const string profileName = "ProfileName";
        const string profileType = "ProfileType";
        const string profileAttributeValues = "ProfileAttributeValues";

        /*
        Values identifying the types of profiles used is provided below.
        */

        const unsigned short AAL1Profile = 1;
        const unsigned short AAL2Profile = 2;
        const unsigned short AAL2PVCProfile = 3;
        const unsigned short AAL5Profile = 4;
        const unsigned short AlarmSeverityAssignmentProfile = 5;
        const unsigned short ATMNetworkAccessProfile = 6;
        const unsigned short BridgedLANServiceProfile = 7;
        const unsigned short CESServiceProfile = 8;
        const unsigned short DS1Profile = 9;
        const unsigned short DS3Profile = 10;
        const unsigned short EthernetProfile = 11;
        const unsigned short IDLCCallProcessingProfile = 12;
        const unsigned short LESProfile = 13;
```

```
        const unsigned short MACBridgeServiceProfile = 14;
        const unsigned short SSCSParameterProfile1 = 15;
        const unsigned short SSCSParameterProfile2 = 16;
        const unsigned short TrafficDescriptorProfile = 17;
        const unsigned short UNIProfile = 18;
        const unsigned short VoiceServiceProfileAAL1 = 19;
        const unsigned short VoiceServiceProfileAAL2 = 20;
        const unsigned short ThresholdData = 21;
        const unsigned short ATMOverbookingFactorProfile = 22;

        /*
        Mapping to filterable data within the structured event is provided for
        a consumer event that involves the creation of a profile object.

        {
        {"ProfileName", any (NameType)},
        {"ProfileType", any (ProfileKindType)},
        {"EventTime", any (GeneralizedTimeType)},
        {"ProfileAttributeValues", any (ProfileStruct)},
        {"NotificationIdentifier", any (NotificationIdentifierType)}
        }

        */


    }; // interface ProfileConsumer


    interface ProfileUsageMgr : itut_x780::ManagedObject {

        // See 9.9.2.1 for the description of the behaviour of this operation

        void reName (
             in NameType oldProfileName,
             in NameType newProfileName)
             raises (UnknownProfiles,
                  AccessDenied,
                  DuplicateProfileName
                  );

        // See 9.9.2.2 for the description of the behaviour of this operation

        boolean inUse (
             in NameType  profileName)
             raises (UnknownProfiles,
             AccessDenied);

        // See 9.9.2.3 for the description of the behaviour of this operation

        void suspendUse (
             in NameType profileName)
             raises (UnknownProfiles,
                  AccessDenied);

        // See 9.9.2.4 for the description of the behaviour of this operation

        void resumeUse (
             in NameType profileName)
             raises (UnknownProfiles, AccessDenied) ;

        // See 9.9.2.5 for the description of the behaviour of this operation

        void deleteProfile (
             in NameType profileName)
```

```
                raises (UnknownProfiles,
                     AccessDenied,
                     ProfileInUse);

     }; // interface ProfileUsageMgr

     /*
     This object is instantiated on the actor called Profile Object Repository.
     The Supplier Management System is the client.
     */


     interface ProfileRetriever : itut_x780::ManagedObject {

          // See 9.9.3.1 for the description of the behaviour of this operation

          ProfileInfoType retrieve (
               in NameType profileName)
               raises (UnknownProfiles);

     }; // interface ProfileRetriever

}; // module ProfileManager

}; // module q834_4
#endif
```

C.10 Q834Registrar.idl

```
#ifndef __Q834_4_REGISTRAR_DEFINED
#define __Q834_4_REGISTRAR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module Registrar
{

     // begin definitions from other idl files


     // From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::SerialNumType SerialNumType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::DCNAddressType DCNAddressType;
     typedef Q834Common::AdministrationDomainType AdministrationDomainType;
     typedef Q834Common::UserLabelSeqType UserLabelSeqType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define EquipmentFailure Q834Common::EquipmentFailure
```

```
#define InsufficientPONBW  Q834Common::InsufficientPONBW
#define InvalidSerialNumSyntax Q834Common::InvalidSerialNumSyntax
#define MaxSubtendingNodesExceeded Q834Common::MaxSubtendingNodesExceeded
#define UnknownNE Q834Common::UnknownNE
#define UnknownPort Q834Common::UnknownPort
#define DCNTimeout Q834Common::DCNTimeout
#define DeniedAccess Q834Common::DeniedAccess
#define BackupInProgress Q834Common::BackupInProgress
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define InvalidUserLabelSyntax Q834Common::InvalidUserLabelSyntax
#define SynchInProgress Q834Common::SynchInProgress


     // End definitions from other idl files

     // Local data types

     // Local exceptions
     exception TooManyNEs {};
     exception InvalidDCNAddress {};
     exception AddressLabelMismatch {};
     exception APONLayerFailure {};
     exception InvalidPort {};
     exception HWServicesMismatch {};

     // End local definitions

     valuetype NERegistrarValueType: itut_x780::ManagedObjectValueType {

          public ManagedEntityIdSeqType NEList; // GET
     };


     interface NERegistrar : itut_x780::ManagedObject {


          // See 9.10.1.1 for the description of the behaviour of this operation

          ManagedEntityIdType registerNE (
               in DCNAddressType nEDCNAddress,
               in UserLabelType nEUserLabel,
               in AdministrationDomainType administrationDomain)
               raises (AccessDenied,
                    DCNTimeout,
                    AddressLabelMismatch,
                    DuplicateUserLabel,
                    TooManyNEs,
                    InvalidDCNAddress,
                    CanNotAssignManagedEntityId,
                    CanNotRetrieveUserLabel,
                    DeniedAccess,
                    InvalidUserLabelSyntax);

          // See 9.10.1.2 for the description of the behaviour of this operation

          void modifyNEDCNAddress (
               in ManagedEntityIdType nEManagedEntityId,
               in DCNAddressType newNEDCNAddress)
               raises(AccessDenied,
                    DeniedAccess,
                    AddressLabelMismatch,
                    DCNTimeout,
                    CommFailure,
                    UnknownNE,
```

```
          InvalidDCNAddress,
          BackupInProgress);


// See 9.10.1.3 for the description of the behaviour of this operation

ManagedEntityIdType rangeONTorONU (
     in ManagedEntityIdType oltManagedEntityId,
     in UserLabelType nEUserLabel,
     in SerialNumType serialNum,
     in ManagedEntityIdType port ) // OLT PON Port
     raises (AccessDenied,
          CommFailure,
          EquipmentFailure,
          UnknownNE,
          UnknownPort,
          MaxSubtendingNodesExceeded,
          InsufficientPONBW,
          InvalidSerialNumSyntax,
          APONLayerFailure,
          DuplicateUserLabel,
          InvalidUserLabelSyntax,
          BackupInProgress,
          SynchInProgress );

// See 9.10.1.4 for the description of the behaviour of this operation

ManagedEntityIdType rangeReplacementNE (
     in ManagedEntityIdType oldNEManagedEntityId,
     in UserLabelType newNEUserLabel,
     in SerialNumType replacementSerialNum )
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          InvalidSerialNumSyntax,
          APONLayerFailure,
          EquipmentFailure,
          InvalidUserLabelSyntax,
          DuplicateUserLabel,
          HWServicesMismatch,
          BackupInProgress,
          SynchInProgress);

// See 9.10.1.5 for the description of the behaviour of this operation

ManagedEntityIdType rangeUpgradeNE (
     in ManagedEntityIdType oldNEManagedEntityId,
     in ManagedEntityIdType newNEManagedEntityId,
     in UserLabelType newNEUserLabel,
     in SerialNumType newNESerialNum )
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          InvalidSerialNumSyntax,
          APONLayerFailure,
          EquipmentFailure,
          InvalidUserLabelSyntax,
          DuplicateUserLabel,
          HWServicesMismatch,
          BackupInProgress,
          SynchInProgress);

// See 9.10.1.6 for the description of the behaviour of this operation
```

```
          ManagedEntityIdType moveONTorONU(
               in ManagedEntityIdType oldNEManagedEntityId,
               in ManagedEntityIdType newPONPort)
               raises (AccessDenied,
                    CommFailure,
                    UnknownPort,
                    APONLayerFailure,
                    EquipmentFailure,
                    InsufficientPONBW,
                    BackupInProgress,
                    SynchInProgress,
                    UnknownNE );

          // See 9.10.1.7 for the description of the behaviour of this operation

          ManagedEntityIdSeqType getSubtendingNEList(
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE, AccessDenied);

          // See 9.10.1.8 for the description of the behaviour of this operation

          ManagedEntityIdSeqType nEListGet ()
               raises (AccessDenied);


          // See 9.10.1.9 for the description of the behaviour of this operation

          void deRegisterNE(
               in ManagedEntityIdType nE)
               raises (UnknownNE,
                    AccessDenied);

          // See 9.10.1.10 for the description of the behaviour of this
          operation

          ManagedEntityIdType associateNE(
in ManagedEntityIdType preProvisionedNE,
               in ManagedEntityIdType discoveredNE)
               raises (UnknownManagedEntity,
                    AccessDenied);

     }; // interface NERegistrar

}; // module Registrar

}; // module q834_4

#endif
```

C.11 Q834ResourceAllocation.idl

```
#ifndef __Q834_4_RESOURCEALLOCATOR_DEFINED
#define __Q834_4_RESOURCEALLOCATOR_DEFINED


#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {
```

```
module ResourceAllocation {

     // Begin definitions from other idl files

     // From Q834Common
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::ReservationIdSeqType ReservationIdSeqType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::EndPointType EndPointType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define ConnectionCountExceeded Q834Common::ConnectionCountExceeded
#define InsufficientBW Q834Common::InsufficientBW
#define MaxSubtendingNodesExceeded Q834Common::MaxSubtendingNodesExceeded
#define UnknownNE Q834Common::UnknownNE
#define UnknownPort Q834Common::UnknownPort
#define UnknownProfiles Q834Common::UnknownProfiles
#define UnknownReservationId Q834Common::UnknownReservationId
#define UnknownServiceInstance Q834Common::UnknownServiceInstance
#define ProfileSuspended Q834Common::ProfileSuspended

     // End of definitions from other idl files

     // Local data types


     struct BandwidthInfoType {
          long upstreamBW;
          long downstreamBW;
     };

     struct PortBandwidthType {
          BandwidthInfoType portBandwidth;
          ManagedEntityIdType portId;
     };

     typedef sequence<PortBandwidthType> PortBandwidthSeqType;

typedef PortBandwidthSeqType AvailableSysBandwidthSeqType;
     typedef PortBandwidthSeqType ReservedBandwidthSeqType;

     struct ReservationInfoType {
          ReservationIdType reservationId;
          EndPointType endPointA;
          EndPointType endPointB;
          NameSeqType profileNameList; // servicetemplates
          ServiceInstanceIdType serviceInstanceId;
          ReservedBandwidthSeqType reservedBandwidth;
     };

     struct ReservationBandwidthType {
          ReservationIdType reservationId;
          ReservedBandwidthSeqType reservedBandwidth;
     };

     // Local exceptions

     // End local definitions


     interface ResourceAllocator : itut_x780::ManagedObject {
```

```
// See 9.11.1.1 for the description of the behaviour of this operation

ReservationBandwidthType reserveForService(
     in EndPointType endPointA,
     in EndPointType endPointZ,
     in NameSeqType networkCharacteristicsProfiles,
     in ServiceInstanceIdType serviceInstanceId )
     raises (UnknownNE,
          UnknownPort,
          UnknownProfiles,
          InsufficientBW,
          MaxSubtendingNodesExceeded,
          ConnectionCountExceeded,
          CommFailure,
          AccessDenied,
          ProfileSuspended );

// See 9.11.1.2 for the description of the behaviour of this operation

AvailableSysBandwidthSeqType cancelReservation (
     in ReservationIdType reservationId )
     raises (UnknownReservationId,
          CommFailure,
          AccessDenied);

// See 9.11.1.3 for the description of the behaviour of this operation

ReservationIdType getReservationId (
     in ServiceInstanceIdType serviceInstanceId)
     raises (UnknownServiceInstance, AccessDenied);

// See 9.11.1.4 for the description of the behaviour of this operation

ReservedBandwidthSeqType reportReservedResources (
     in ManagedEntityIdType nEManagedEntityId)
     raises (UnknownNE, AccessDenied);

// See 9.11.1.5 for the description of the behaviour of this operation

ReservationIdSeqType getReservations(
     in ManagedEntityIdType nEManagedEntityId)
     raises (UnknownNE, AccessDenied);

// See 9.11.1.6 for the description of the behaviour of this operation

AvailableSysBandwidthSeqType cancelAllRemainingReservations(
     in ManagedEntityIdType nEManagedEntityId)
     raises (UnknownNE,
          CommFailure,
          AccessDenied);

// See 9.11.1.7 for the description of the behaviour of this operation

ReservationInfoType getReservation (
     in ReservationIdType reservationId)
     raises (UnknownReservationId,
          AccessDenied);

// See 9.11.1.8 for the description of the behaviour of this operation

AvailableSysBandwidthSeqType getAvaliableSysBandwidth(
     in ManagedEntityIdType nEManagedEntityId)
```

```
                raises (UnknownNE,
                     CommFailure,
                     AccessDenied);

     }; // interface ResourceAllocator


}; // module ResourceAllocation

}; // module q834_4

#endif
```

C.12 Q834SchedulerManagement.idl

```
#ifndef __Q834_4_SCHEDULERMGR_DEFINED
#define __Q834_4_SCHEDULERMGR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module SchedulerManagement {
     // Begin definitions from other idl files


     // From Q834Common

     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::OperationalStateType OperationalStateType;

#define AccessDenied Q834Common::AccessDenied
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidStartTime Q834Common::InvalidStartTime
#define InvalidStopTime Q834Common::InvalidStopTime

     // End definitions from other idl files

     // Local data types


     enum HourlyDailyWeeklyMonthlyIndType {
          Hourly,
          Daily,
          Weekly,
          Monthly
     };

     enum DayofWeekType {
          Sunday,
          Monday,
          Tuesday,
          Wednesday,
          Thursday,
          Friday,
          Saturday,
          Unspecified
```

```
};

typedef short DayOfMonthType; // 0 is interpreted to mean unspecified.


struct TriggerTimeType
{
     long time; // trigger time in number of seconds
     DayofWeekType dayOfWeek;
     DayOfMonthType dayofMonth;
};

typedef sequence<TriggerTimeType> TriggerTimeMatrixSeqType;

struct SchedulerType
{
     UserLabelType schedulerName;
     GeneralizedTimeType startTime; // schedule start time
     GeneralizedTimeType stopTime; // schedule stop time
     HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd;
     TriggerTimeMatrixSeqType matrix;
     OperationalStateType operationalState;
     AdministrativeStateType administrativeState;
};

typedef sequence<SchedulerType> SchedulerSeqType;

// Local exceptions

exception MatrixSchedulerTypeMismatch {};
exception InvalidTrigger {};
exception ScheduleInUse {};

// End local definitions


valuetype SchedulerMgrValueType: itut_x780::ManagedObjectValueType {

     public SchedulerSeqType schedulerList; // GET

};

interface SchedulerMgr : itut_x780::ManagedObject {


     // See 9.12.1.1 for the description of the behaviour of this operation

     void makeScheduler (
          in UserLabelType schedulerName,
          in GeneralizedTimeType startTime,
          in GeneralizedTimeType stopTime,
          in HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd,
          in TriggerTimeMatrixSeqType matrix)
          raises (InvalidStartTime,
               InvalidStopTime,
               DuplicateUserLabel,
               MatrixSchedulerTypeMismatch,
               AccessDenied,
               InvalidTrigger );


     // See 9.12.1.2 for the description of the behaviour of this operation

     void suspendScheduler (
```

```
            in UserLabelType schedulerName)
            raises (UnknownScheduler,
                AccessDenied );

        // See 9.12.1.3 for the description of the behaviour of this operation

        void resumeScheduler (in UserLabelType schedulerName) raises
(UnknownScheduler, AccessDenied );


        // See 9.12.1.4 for the description of the behaviour of this operation

        void modifyTime (
            in UserLabelType schedulerName,
            in GeneralizedTimeType newStartTime,
            in GeneralizedTimeType newStopTime)
            raises (InvalidStartTime,
                InvalidStopTime,
                UnknownScheduler,
                AccessDenied );


        // See 9.12.1.5 for the description of the behaviour of this operation

        void changeSchedulerName (
            in UserLabelType oldSchedulerName,
            in UserLabelType newSchedulerName)
            raises (UnknownScheduler,
                DuplicateUserLabel,
                AccessDenied );

        // See 9.12.1.6 for the description of the behaviour of this operation

        void modifyTriggerTimes (
            in UserLabelType schedulerName,
            in HourlyDailyWeeklyMonthlyIndType newHourlyDailyWeeklyMonthly,
            in TriggerTimeMatrixSeqType newMatrix)
            raises (UnknownScheduler,
                MatrixSchedulerTypeMismatch,
                AccessDenied,
                InvalidTrigger );

        // See 9.12.1.7 for the description of the behaviour of this operation

        void removeScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler,
                AccessDenied,
                ScheduleInUse );

        // See 9.12.1.8 for the description of the behaviour of this operation

        SchedulerType retrieveScheduler (in UserLabelType schedulerName)
            raises (UnknownScheduler,
                AccessDenied) ;

        // See 9.12.1.9 for the description of the behaviour of this operation

        SchedulerSeqType schedulerListGet () raises (AccessDenied);

    }; // interface SchedulerMgr

}; // module SchedulerManagement
```

```
}; // module q834_4

#endif
```

C.13 Q834ServiceProvisioning.idl

```
#ifndef __Q834_4_SERVICEPROVISIONING_DEFINED
#define __Q834_4_SERVICEPROVISIONING_DEFINED


#include "Q834Common.idl"
#include "Q834ProfileManager.idl"

#pragma prefix "itu.Int"

module q834_4 {

module ServiceProvisioning {

// Begin definitions from other idl files

// From Q834Common
     typedef Q834Common::RDNType RDNType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::EndPointType EndPointType;
     typedef Q834Common::NameType NameType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define ConnectionCountExceeded Q834Common::ConnectionCountExceeded
#define EquipmentFailure Q834Common::EquipmentFailure
#define InsufficientBW Q834Common::InsufficientBW
#define InsufficientPONBW  Q834Common::InsufficientPONBW
#define UnknownProfiles Q834Common::UnknownProfiles
#define UnknownReservationId  Q834Common::UnknownReservationId
#define UnknownNE Q834Common::UnknownNE
#define UnknownServiceInstance Q834Common::UnknownServiceInstance
#define ProfileSuspended ProfileManager::ProfileSuspended
#define InvalidStartTime Q834Common::InvalidStartTime
#define InvalidStopTime Q834Common::InvalidStopTime
#define ParameterViolation Q834Common::ParameterViolation
#define UnknownPort Q834Common::UnknownPort
#define ProfileSuspended Q834Common::ProfileSuspended


// End definitions from other idl files

// Local data types

// Local exceptions
     exception UnknownConnection {};
     exception ConnectionAlreadyExists {};

// End local definitions


     interface ServiceProvisioner : itut_x780::ManagedObject {
```

```
// See 9.13.1.1 for the description of the behaviour of this operation

ManagedEntityIdType provisionConnection(
     in EndPointType endPointA,
     in EndPointType endPointB,
     in NameSeqType networkCharacteristicsProfiles,
     in ServiceInstanceIdType serviceInstanceId,
     in AdministrativeStateType administrativeState)
     raises (UnknownNE,
          UnknownProfiles,
          UnknownPort,
          InsufficientBW,
          ConnectionCountExceeded,
          CommFailure,
          EquipmentFailure,
          ParameterViolation,
          AccessDenied,
          InsufficientPONBW,
          ConnectionAlreadyExists,
          ProfileSuspended);

// See 9.13.1.2 for the description of the behaviour of this operation

ManagedEntityIdType provisionReservation(
     in ReservationIdType reservationId,
     in AdministrativeStateType administrativeState)
     raises (UnknownReservationId,
          AccessDenied);

// See 9.13.1.3 for the description of the behaviour of this operation

void deleteConnection (
     in ManagedEntityIdType subnetworkConnectionId)
     raises (UnknownConnection,
          CommFailure,
          EquipmentFailure,
          AccessDenied);

// See 9.13.1.4 for the description of the behaviour of this operation

ManagedEntityIdType modifyConnection (
     in ManagedEntityIdType subnetworkConnectionId,
     in ManagedEntityIdType portB,
     in NameSeqType newNetworkCharacteristicsProfiles)
     raises (UnknownConnection,
          UnknownPort,
          UnknownProfiles,
          InsufficientBW,
          AccessDenied,
          ProfileSuspended);

// See 9.13.1.5 for the description of the behaviour of this operation

void suspendService (
     in ServiceInstanceIdType serviceInstanceId,
     in GeneralizedTimeType startTime,
     in GeneralizedTimeType stopTime)
     raises (UnknownServiceInstance,
          InvalidStartTime,
          InvalidStopTime,
          AccessDenied);

// See 9.13.1.6 for the description of the behaviour of this operation
```

```
          void resumeService(
               in ServiceInstanceIdType serviceInstanceId)
               raises (UnknownServiceInstance,
                    AccessDenied);

     }; // interface ServiceProvisioner

}; // module ServiceProvisioning

}; // module q834_4

#endif
```

C.14 Q834Synchroniser.idl

```
#ifndef __Q834_4_SYNCHRONISER_DEFINED
#define __Q834_4_SYNCHRONISER_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module Synchroniser
{

// begin definitions from other idl files


// From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::UserLabelTypeUserLabelType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define DCNTimeout Q834Common::DCNTimeout
#define EquipmentFailure Q834Common::EquipmentFailure
#define UnknownNE Q834Common::UnknownNE
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidScheduler Q834Common::InvalidScheduler
#define BackupInProgress Q834Common::BackupInProgress
#define Timeout Q834Common::Timeout
#define SynchInProgress Q834Common::SynchInProgress


// End definitions from other idl files

// Local data types

     typedef sequence<short> CurrentListingSeqType;

     struct ScheduledSynchNEType {
          ManagedEntityIdType managedEntityId;
          UserLabelType schedulerName;
     };
     typedef sequence<ScheduledSynchNEType> ScheduledSynchNESeqType;

// Local exceptions
     exception NoSynchInProgress {};
```

```
// End local definitions

     valuetype SynchroniserValueType: itut_x780::ManagedObjectValueType {

          public ScheduledSynchNESeqType scheduledSynchNEList; // GET

     };


     interface NESynchroniser : itut_x780::ManagedObject {

          // Define constants for current event listings

          const short CURRENT_ALARM = 1;
          const short CURRENT_OPERATIONAL_STATE_DISABLED = 2;
          const short CURRENT_ADMINSTRATIVE_STATE_LOCKED = 3;
          const short CURRENT_PROTECTION_SWITCHING_EVENT = 4;
          const short CURRENT_SERVICE_OUTAGE = 5;

          // See 9.14.1.1 for the description of the behaviour of this operation

          void synchNE(
               in ManagedEntityIdType nEManagedEntityId)
               raises (AccessDenied,
                    CommFailure,
                    UnknownNE,
                    EquipmentFailure,
                    BackupInProgress,
                    SynchInProgress);

          // See 9.14.1.2 for the description of the behaviour of this operation


          void abortSynchNE(
               in ManagedEntityIdType nEManagedEntityId)
               raises (AccessDenied,
                    CommFailure,
                    UnknownNE,
                    EquipmentFailure,
                    NoSynchInProgress);

          // See 9.14.1.3 for the description of the behaviour of this operation

          void scheduleSynchNE(
               in ManagedEntityIdType nEManagedEntityId,
               in UserLabelType schedulerName)
               raises (AccessDenied,
                    UnknownNE,
                    UnknownScheduler,
                    InvalidScheduler);

          // See 9.14.1.4 for the description of the behaviour of this operation

          void modifyNESynchSchedule(
               in ManagedEntityIdType nEManagedEntityId,
               in UserLabelType newSchedulerName)
               raises (AccessDenied,
                    UnknownNE,
                    UnknownScheduler,
                    InvalidScheduler);

          // See 9.14.1.5 for the description of the behaviour of this operation
```

```
          void cancelScheduledSynchNE(
               in ManagedEntityIdType nEManagedEntityId)
               raises (AccessDenied,
                    UnknownNE);

          // See 9.14.1.6 for the description of the behaviour of this operation

          void synchCurrentEventListings(
               in ManagedEntityIdType nEManagedEntityId,
               in CurrentListingSeqType currentListingTypeList)
               raises (AccessDenied,
                    CommFailure,
                    DCNTimeout,
                    UnknownNE,
                    EquipmentFailure,
                    Timeout);

          // See 9.14.1.7 for the description of the behaviour of this operation

          ScheduledSynchNESeqType scheduledSynchNEListGet ()
               raises (AccessDenied);

     }; // interface NESynchroniser

}; // module Synchroniser

}; // module q834_4

#endif
```

C.15 Q834Test.idl

```
#ifndef __Q834_4_TEST_DEFINED
#define __Q834_4_TEST_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module Test
{

     // begin definitions from other idl files

     // From Q834Common

     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::LoopbackLocationIdSeqType
     LoopbackLocationIdSeqType;
     typedef Q834Common::StatusValueType StatusValueType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::TrackingObjectIdType TrackingObjectIdType;

#define AccessDenied Q834Common::AccessDenied
```

```
#define CommFailure Q834Common::CommFailure
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidScheduler Q834Common::InvalidScheduler
#define InvalidStopTime Q834Common::InvalidStopTime
#define InvalidStartTime Q834Common::InvalidStartTime
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define UnknownNE Q834Common::UnknownNE
#define UnknownServiceInstance Q834Common::UnknownServiceInstance

     // End definitions from other idl files


     // Local data types

     enum DirectionalityType {
             Egress,
          Ingress,
          BothDirections
     };

     struct ATMLoopbackInfoType {
          DirectionalityType directionality;
          LoopbackLocationIdSeqType targetLLID;
          boolean segmentCellInd;
     };

     typedef   unsigned short TestIterationNumType;

     struct ATMLoopbackResultType {
          LoopbackLocationIdSeqType loopbackingLLID;
          unsigned long responseTime; //in microseconds
          boolean succeeded; //true or false
     };

     typedef sequence<ATMLoopbackResultType> ATMLoopbackResultSeqType;

     struct AggregateATMLoopbackResultType {
          unsigned short iterationSeqNum;
          ATMLoopbackResultSeqType iterationTestResults;
     };

     typedef sequence<AggregateATMLoopbackResultType>
     AggregateATMLoopbackResultSeqType;

     struct ATMContinuityCheckInfoType {
          DirectionalityType directionality;
          boolean segmentCellInd;
     };


     typedef short DiagnosticType; // Supplier specific.
     typedef DiagnosticType ResourceSelfTestInfoType;
     typedef sequence<ResourceSelfTestInfoType> ResourceSelfTestInfoSeqType;

     struct ResourceSelfTestResultType {
             DiagnosticType diagnostic;
             boolean testPassed;
     };

     typedef sequence<ResourceSelfTestResultType>
     ResourceSelfTestResultSeqType;
     typedef long long CCSetUpIdType;
     typedef TrackingObjectIdType TestTrackingObjectIdType;
     typedef long long LoopbackTrackingObjectIdType;
```

```
     typedef unsigned short LoopbackTestType;

     struct LoopbackInfoType {
          LoopbackTrackingObjectIdType trackingId;
          unsigned long remainingTime; // in seconds
ManagedEntityIdType ctpId;
          LoopbackTestType loopbackTest;
          DirectionalityType directionality;
     };

     typedef sequence<LoopbackInfoType> LoopbackInfoSeqType;

     typedef string SupportedTestType;

     typedef sequence<SupportedTestType> SupportedTestSeqType;

     struct ScheduledTestNEType {
            ManagedEntityIdType managedEntityId;
            SupportedTestType supportedTest;
            UserLabelType schedulerName;
     };
     typedef sequence<ScheduledTestNEType> ScheduledTestNESeqType;

     struct TestHistoryType {
          ManagedEntityIdType managedEntityId;
          SupportedTestType supportedTest;
          StatusValueType testStatus;
     };
     typedef sequence<TestHistoryType> TestHistorySeqType;

     // Local exceptions

     exception InvalidTimeoutPeriod {};
     exception NotAvailableForTest {};
     exception InvalidLocationId {};
     exception UnknownTest {};
     exception UncontrolledTestInProgress {};
     exception InvalidDirection {};
     exception InvalidTestOperations {};

     // End local definitions

     valuetype TestActionPerformerValueType: itut_x780::ManagedObjectValueType {

          public ScheduledTestNESeqType scheduledTestNEList; // GET

     };


     interface TestActionPerformer : itut_x780::ManagedObject {

          // See 9.15.1.1 for the description of the behaviour of this operation

          AggregateATMLoopbackResultSeqType aTMLoopback (
               in UserIdType testRequestorId,
               in ManagedEntityIdType ctp,
               in ATMLoopbackInfoType aTMLoopbackInfo,
               in TestIterationNumType testIterationNum,
               in ServiceInstanceIdType serviceInstanceId)
               raises (AccessDenied,
                    CommFailure,
                    UnknownManagedEntity,
```

```
          NotAvailableForTest,
          InvalidLocationId,
          InvalidDirection);

// See 9.15.1.2 for the description of the behaviour of this operation

CCSetUpIdType initializeContinuityCheck(
     in UserIdType testRequestorId,
     in ManagedEntityIdType sourceCtp,
     in ATMContinuityCheckInfoType aTMContinuityCheckInfo,
     in GeneralizedTimeType stopTime,
     in ServiceInstanceIdType serviceInstanceId)
     raises (AccessDenied,
          CommFailure,
          UnknownManagedEntity,
          NotAvailableForTest,
          InvalidStartTime,
          InvalidStopTime,
          InvalidDirection);

// See 9.15.1.3 for the description of the behaviour of this operation

void terminateContinuityCheck(
     in CCSetUpIdType cCSetUpId)
     raises (AccessDenied,
          CommFailure,
          UnknownTest);

// See 9.15.1.4 for the description of the behaviour of this operation

TestTrackingObjectIdType scheduleResourceSelfTest(
     in UserIdType testRequestorId,
     in ManagedEntityIdType targetNE,
     in unsigned long timeOutPeriod, //In seconds.
     in ResourceSelfTestInfoSeqType specificTestInfo,
     in UserLabelType schedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownScheduler,
          InvalidScheduler,
          InvalidTimeoutPeriod,
          InvalidTestOperations);

// See 9.15.1.5 for the description of the behaviour of this operation

         void modifyResourceSelfTestSchedule(
           in TestTrackingObjectIdType testTrackingObjectId,
           in UserLabelType newSchedulerName)
           raises (AccessDenied,
                UnknownTest,
                UnknownScheduler,
                InvalidScheduler);

// See 9.15.1.6 for the description of the behaviour of this operation

     void cancelScheduledResourceSelfTest (
          in TestTrackingObjectIdType testTrackingObjectId)
          raises (AccessDenied,
                UnknownTest);

// See 9.15.1.7 for the description of the behaviour of this operation

TestTrackingObjectIdType conductResourceSelfTest (
     in UserIdType testRequestorId,
```

```
        in ManagedEntityIdType targetNE,
        in unsigned long timeOutPeriod, //In seconds.
        in ResourceSelfTestInfoSeqType specificTestInfo)
        raises (AccessDenied,
             CommFailure,
             UnknownNE,
             InvalidTimeoutPeriod,
             InvalidTestOperations);

    // See 9.15.1.8 for the description of the behaviour of this operation

    ResourceSelfTestResultSeqType terminateResourceSelfTest (
        in TestTrackingObjectIdType testTrackingObjectId)
        raises (UnknownTest,
             UncontrolledTestInProgress,
             AccessDenied);

    // See 9.15.1.9 for the description of the behaviour of this operation

    LoopbackTrackingObjectIdType initiateLoopback (
        in UserIdType testRequestorId,
        in ManagedEntityIdType loopingCtp,
        in long duration, //In minutes.
        in DirectionalityType directionality,
        in LoopbackTestType loopbackTest,
        in ServiceInstanceIdType serviceInstanceId)
        raises (AccessDenied,
             CommFailure,
             UnknownManagedEntity,
             NotAvailableForTest);

    // See 9.15.1.10 for the description of the behaviour of this
    operation

    void terminateLoopback(
        in LoopbackTrackingObjectIdType testTrackingObjectId)
        raises (UnknownTest,
             AccessDenied);

    // See 9.15.1.11 for the description of the behaviour of this
    operation

    LoopbackInfoType getLoopbackInfo(
        in ManagedEntityIdType cTP)
        raises (UnknownManagedEntity,
             AccessDenied);

    // See 9.15.1.12 for the description of the behaviour of this
    operation

    LoopbackInfoSeqType getLoopbackInfoByNE(
        in ManagedEntityIdType nEId)
        raises (UnknownManagedEntity,
             AccessDenied);

    // See 9.15.1.13 for the description of the behaviour of this
    operation

    StatusValueType getTestStatus (
        in LoopbackTrackingObjectIdType id)
        raises (AccessDenied,
             UnknownTest);

    // See 9.15.1.14 for the description of the behaviour of this
```

```
          operation

          ScheduledTestNESeqType  scheduledTestNEListGet()
               raises (AccessDenied);

          // See 9.15.1.15 for the description of the behaviour of this
          operation

          TestHistorySeqType testHistoryByManagedEntity (
               in ManagedEntityIdType managedEntityId)
               raises (UnknownManagedEntity,
                    AccessDenied);

          // See 9.15.1.16 for the description of the behaviour of this
          operation

          TestHistorySeqType testHistoryByServiceInstance (
               in ServiceInstanceIdType serviceInstanceId)
               raises (UnknownServiceInstance,
                    AccessDenied);

     }; // interface TestActionPerformer

}; // module Test

}; // module q834_4

#endif
```

C.16 Q834Filetransfer.idl

```
#ifndef __Q834_4_TRANSFERMGR_DEFINED
#define __Q834_4_TRANSFERMGR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module FileTransfer
{

// begin definitions from other idl files

// From Q834Common

     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::StatusValueType StatusValueType;
     typedef Q834Common::DCNAddressType DCNAddressType;
     typedef Q834Common::FilenameType FilenameType;
     typedef Q834Common::TransferTrackingObjectIdType
     TransferTrackingObjectIdType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define UnknownScheduler Q834Common::UnknownScheduler
#define UnknownDestinationServer Q834Common::UnknownDestinationServer
```

```
#define InvalidScheduler Q834Common::InvalidScheduler
#define UnknownRecordSet Q834Common::UnknownRecordSet

// End definitions from other idl files

// Local data types



     struct FileTransferHistoryType {
       ManagedEntityIdType recordSetId;
       DCNAddressType destinationServerAddr;
       UserIdType  userId;
       FilenameType destinationFile;
       GeneralizedTimeType transferTime;
     };
     typedef sequence<FileTransferHistoryType> FileTransferHistorySeqType;

     struct ScheduledFileTransferType {
       ManagedEntityIdType recordSetId;
       UserLabelType schedulerName;
     };
     typedef sequence<ScheduledFileTransferType> ScheduledFileTransferSeqType;

// Local exceptions

     exception UnknownTransferProcess {};

// End local definitions


     valuetype TransferMgrValueType: itut_x780::ManagedObjectValueType {

          public FileTransferHistorySeqType fileTransferHistoryList; // GET
          public ScheduledFileTransferSeqType scheduledFileTransferList; // GET

     };

     interface TransferMgr : itut_x780::ManagedObject {

          // See 9.16.1.1 for the description of the behaviour of this operation

               TransferTrackingObjectIdType fileTransfer(
                    in ManagedEntityIdType recordSetId,
                    in DCNAddressType destinationServerAddr,
                    in UserIdType userId,
                    in PasswordType password,
                    in FilenameType destinationFile,
                    in boolean overwriteExistingFile)
                    raises (AccessDenied,
                         CommFailure,
                         UnknownRecordSet,
                         UnknownDestinationServer
                         );


          // See 9.16.1.2 for the description of the behaviour of this operation

               TransferTrackingObjectIdType scheduleFileTransfer(
                    in ManagedEntityIdType recordSetId,
                    in DCNAddressType destinationServerAddr,
                    in UserIdType userId,
                    in PasswordType password,
                    in FilenameType destinationFile,
```

```
                    in boolean overwriteExistingFile,
                    in UserLabelType schedulerName)
                    raises (AccessDenied,
                         UnknownRecordSet,
                         UnknownDestinationServer,
                         UnknownScheduler,
                         InvalidScheduler);

          // See 9.16.1.3 for the description of the behaviour of this operation

                  void modifyFileTransferSchedule(
                    in TransferTrackingObjectIdType transferTrackingObjectId,
                    in UserLabelType newSchedulerName)
                    raises (AccessDenied,
                         UnknownTransferProcess,
                         UnknownScheduler,
                         InvalidScheduler);

          // See 9.16.1.4 for the description of the behaviour of this operation

               void cancelScheduledFileTransfer (
                    in TransferTrackingObjectIdType transferTrackingObjectId)
                    raises (AccessDenied,
                         UnknownTransferProcess);

          // See 9.16.1.5 for the description of the behaviour of this operation

               StatusValueType getStatus(
                    in TransferTrackingObjectIdType transferTrackingObjectId)
                    raises (UnknownTransferProcess,
                         AccessDenied);

          // See 9.16.1.6 for the description of the behaviour of this operation

          FileTransferHistorySeqType fileTransferHistoryListGet ()
               raises (AccessDenied);

          // See 9.16.1.7 for the description of the behaviour of this operation

          ScheduledFileTransferSeqType scheduledFileTransferListGet()
               raises (AccessDenied);

     }; // interface TransferMgr

}; // module FileTransfer

}; // module q834_4

#endif
```

Anexo D

Ejemplo de plantillas de punto extremo

Los cuadros D.1 y D.2 proporcionan ejemplos de plantillas de definición de un punto extremo por tipo de servicio. El cuadro D.1 muestra los ejemplos correspondientes al caso en el que el punto extremo forma parte de un puerto NNI. El cuadro D.2 muestra ejemplos correspondientes al caso en el que el punto extremo forma parte de un puerto UNI.

Cuadro D.1/Q.834.4 – Puntos extremos de puerto NNI

Tipo de servicio	Puerto	Parámetro	Perfiles
DS1	TDM DS3	N.º de canal	–
DS3	TDM DS3	-	–
DS1	ATM DS3	VPI/VCI	–
DS3	ATM DS3	VPI/VCI	–
DS1	TDM OC-n	N.º de STS/ N.º de VT1.5	–
DS1	ATM OC-n	VPI/VCI	-
DS3	ATM OC-n	VPI/VCI	-
LAN puenteada	ATM OC-n	VPI/VCI	-
LAN puenteada	ATM DS3	VPI/VCI	-
LAN puenteada	TDM DS3	N.º de canal	-
LAN puenteada	TDM OC-n	N.º de STS/ N.º de VT1.5	-
Voz	ATM DS3	VPI/VCI CID	TrafficDescriptorProfile (ingress) [perfil descriptor de tráfico (entrada)] TrafficDescriptorProfile (egress) [perfil descriptor de tráfico (salida)]
Voz	–	N.º del grupo de interfaz Valor de referencia de llamada	-
ATM	ATM	VPI/VCI	TrafficDescriptorProfile (ingress) [perfil descriptor de tráfico (entrada)] TrafficDescriptorProfile (egress) [perfil descriptor de tráfico (salida)]

Cuadro D.2/Q.834.4 – Puntos extremos de puerto UNI

Tipo de servicio	Puerto	Parámetro	Perfiles
DS1	TDM DS3	N.º de canal	DS1Profile (perfil DS1) CESProfile (perfil CES) AAL1Profile (perfil AAL1)
DS3	TDM DS3	-	DS3Profile (perfil DS3) CESProfile (perfil CES) AAL1Profile (perfil AAL1)
DS1	ATM DS3	VPI/VCI	-
DS3	ATM DS3	VPI/VCI	-
DS1	TDM OC-n o STS-n	N.º de STS/ N.º de VT1.5	AAL1Profile (perfil AAL1) CESProfile (perfil CES)
DS1	ATM OC-n	VPI/VCI	-
DS3	ATM OC-n	VPI/VCI	-
LAN puenteada	Ethernet	N.º de puerto lógico	BridgedLANServiceProfile (perfil de servicio de LAN puenteada) AAL5Profile (perfil AAL5) o AAL1Profile (perfil AAL1)
Voz	RJ-11	-	VoiceServiceProfileAAL2 (perfil de servicio vocal AAL2) SSCSParameterProfile1 (perfil 1 de parámetro SSCS) SSCSParameterProfile2 (perfil 2 de parámetro SSCS) LESService (servicio LES)
Voz	RJ-11	-	VoiceServiceProfileAAL1 (perfil de servicio vocal AAL1)
ATM	ATM	VPI/VCI	TrafficDescriptorProfile (ingress)[perfil descriptor de tráfico (entrada)] TrafficDescriptorProfile (egress)[perfil descriptor de tráfico (salida)]
VLAN	Ethernet	VLAN tag	-

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, 2004