UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T M.3120

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

Enmienda 1
(05/2002)

SERIE M: RGT Y MANTENIMIENTO DE REDES: SISTEMAS DE TRANSMISIÓN, CIRCUITOS TELEFÓNICOS, TELEGRAFÍA, FACSÍMIL Y CIRCUITOS ARRENDADOS INTERNACIONALES

Red de gestión de las telecomunicaciones

Modelo genérico de información a nivel de red y de elemento de red basado en arquitectura de intermediario de petición de objeto común

Enmienda 1: Conmutación de protección

Recomendación UIT-T M.3120 (2001) – Enmienda 1

RECOMENDACIONES UIT-T DE LA SERIE M

RGT Y MANTENIMIENTO DE REDES: SISTEMAS DE TRANSMISIÓN, CIRCUITOS TELEFÓNICOS, TELEGRAFÍA, FACSÍMIL Y CIRCUITOS ARRENDADOS INTERNACIONALES

Introducción y principios generales de mantenimiento y organización del mantenimiento	M.10–M.299
Sistemas internacionales de transmisión	M.300–M.559
Circuitos telefónicos internacionales	M.560–M.759
Sistemas de señalización por canal común	M.760–M.799
Circuitos internacionales utilizados para transmisiones de telegrafía y de telefotografía	M.800–M.899
Enlaces internacionales arrendados en grupo primario y secundario	M.900–M.999
Circuitos internacionales arrendados	M.1000–M.1099
Sistemas y servicios de telecomunicaciones móviles	M.1100–M.1199
Red telefónica pública internacional	M.1200–M.1299
Sistemas internacionales de transmisión de datos	M.1300–M.1399
Designaciones e intercambio de información	M.1400–M.1999
Red de transporte internacional	M.2000–M.2999
Red de gestión de las telecomunicaciones	**M.3000–M.3599**
Redes digitales de servicios integrados	M.3600–M.3999
Sistemas de señalización por canal común	M.4000–M.4999

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

Recomendación UIT-T M.3120

Modelo genérico de información a nivel de red y de elemento de red basado en arquitectura de intermediario de petición de objeto común

Enmienda 1
Conmutación de protección

Resumen

Esta enmienda introduce mejoras a la conmutación de protección en el modelo genérico de información a nivel de red. El modelo describe clases de objeto gestionado de conmutación de protección y las propiedades de las mismas que son genéricas y de utilidad para describir información intercambiada a través de todas las interfaces definidas en la arquitectura RGT M.3010. Estas clases de objeto gestionado genéricas están destinadas a ser aplicables con diferentes tecnologías, arquitecturas y servicios. Las clases de objeto gestionado de conmutación de protección de esta enmienda pueden especializarse para que soporten la gestión de diversas redes de telecomunicaciones.

Orígenes

La enmienda 1 a la Recomendación UIT-T M.3120 (2001), preparada por la Comisión de Estudio 4 (2001-2004) del UIT-T, fue aprobada por el procedimiento de la Resolución 1 de la AMNT el 29 de mayo de 2002.

PREFACIO

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

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

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

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

NOTA

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

PROPIEDAD INTELECTUAL

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

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

© UIT 2002
Reservados todos los derechos. Ninguna parte de esta publicación puede reproducirse por ningún procedimiento sin previa autorización escrita por parte de la UIT.

ÍNDICE

Recomendación UIT-T M.3120

Modelo genérico de información a nivel de red y de elemento de red basado en arquitectura de intermediario de petición de objeto común

Enmienda 1
Conmutación de protección

1 Alcance

Esta enmienda introduce mejoras del modelo de conmutación de protección en el modelo de información a nivel de red y de elemento de red basado en CORBA. El modelo IDL describe clases de objeto gestionado de conmutación de protección y las propiedades de las mismas que son genéricas y de utilidad para describir información intercambiada a través de todas las interfaces definidas en la arquitectura RGT M.3010. Estas clases de objeto de conmutación de protección genéricas están destinadas a ser aplicables con diferentes tecnologías, arquitecturas y servicios. Las distintas industrias de telecomunicación pueden ampliarlas para la gestión de determinadas tecnologías de red, como ATM, SONET/SDH y Ethernet.

2 Referencias

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

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

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

[3] 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*.

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

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

[6] Recomendación UIT-T M.3100 (1995), *Modelo genérico de información de red más* Enmienda 2 (2000).

[7] Recomendación UIT-T G.774 (2001), *Jerarquía digital síncrona – Modelo de información de gestión desde el punto de vista de los elementos de red.*

[8] Recomendación UIT-T G.774.3 (2001), *Jerarquía digital síncrona – Gestión de la protección de secciones de multiplexión desde el punto de vista de los elementos de red.*

3 Visión general del modelo de información de conmutación de protección

En la cláusula 4 se define un conjunto de interfaces CORBA IDL para el modelo de información de conmutación de protección. Estas interfaces se traducen manualmente a partir de un conjunto de clases de objetos gestionados GDMO enmienda 2/M.3100 y G.774.3 siguiendo el marco y las directrices CORBA de la RGT especificados en las Recomendaciones UIT-T Q.816 y X.780 para una interfaz CORBA de granularidad fina.

Además de las interfaces de granularidad fina mencionadas en 4.5, en 5.6 se define un conjunto de interfaces de fachada que las acompañan. Estas interfaces de fachada se definen de conformidad con el marco y las directrices de granularidad gruesa indicadas en las Recomendaciones UIT-T Q.816.1 y X.780.1 para el soporte de la interfaz CORBA de granularidad gruesa. Los nombres de estas interfaces de fachada son los nombres de las correspondientes interfaces de granularidad fina con "_F" (un carácter de subrayado seguido de una "F" mayúscula) añadido al final.

El IDL de esta enmienda es parte integrante de la Rec. UIT-T M.3120, lo que implica que todas las definiciones (clases de objeto, tipo, estructuras, etc.) recogidas en la Rec. UIT-T M.3120 están en el mismo módulo IDL y puede hacerse referencia a las mismas sin el identificador de módulo.

El IDL de esta enmienda se ha compilado satisfactoriamente el IDL sin errores de sintaxis. El compilador utilizado pretende la conformidad con CORBA 2.3, que incluye capacidades de macros de tipos de valor y M4.

Las figuras 1 y 2 muestran las relaciones de herencia, contenencia y asociación de las interfaces CORBA definidas en esta Recomendación. Obsérvese que las interfaces de fachada siguen la misma relación de jerarquía que las correspondientes interfaces de granularidad fina. Debe también señalarse que en las figuras no se muestran interfaces adicionales, aunque son necesarias. Ejemplos son las clases de fábrica.

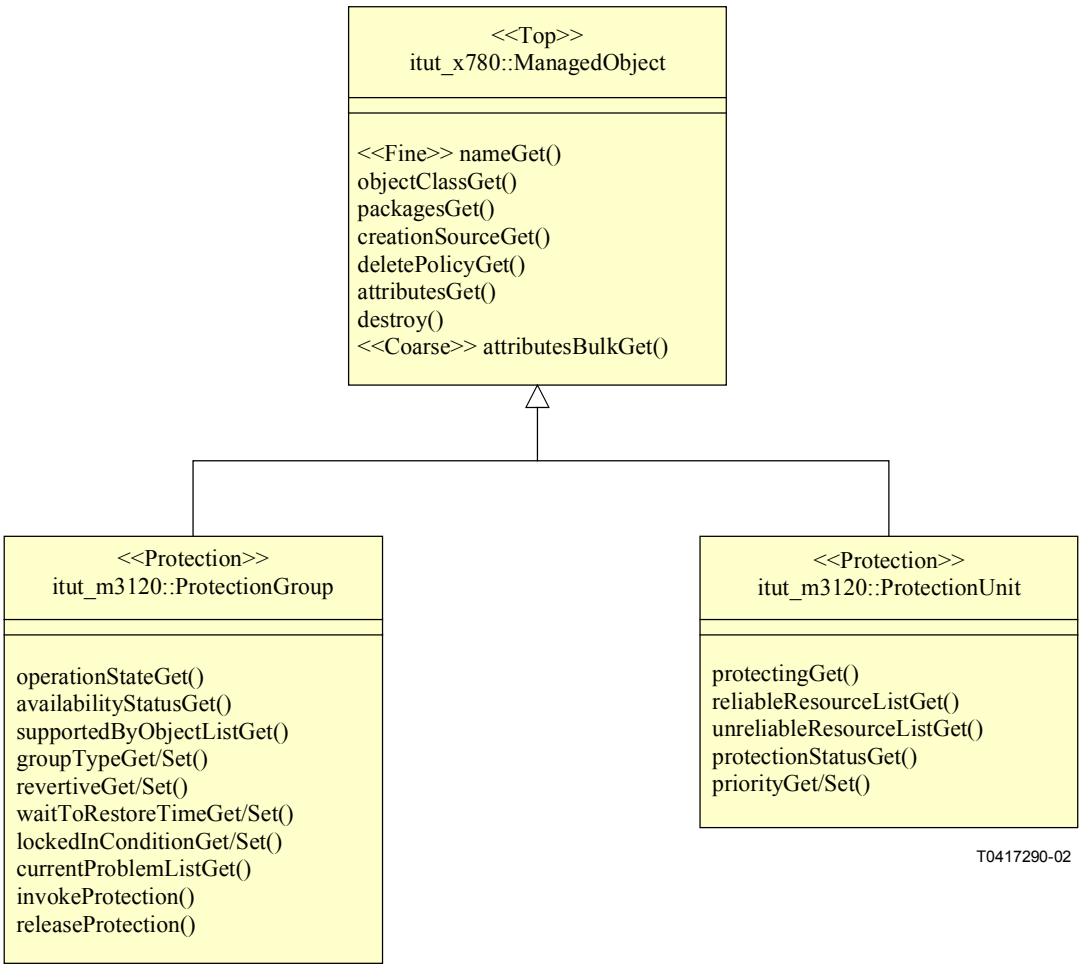

Figura 1/M.3120 – Relación de herencia

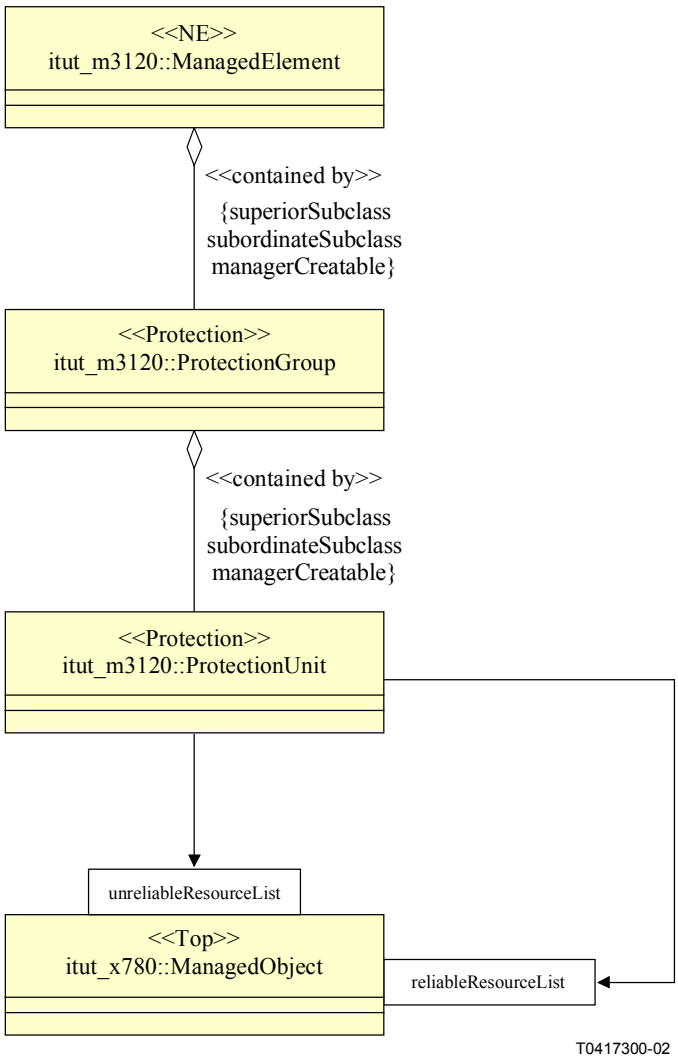

Figura 2/M.3120 – Relaciones de contenencia y de asociación

4 IDL del modelo de información

```
#ifndef _itut_m3120_amd1_idl_
#define _itut_m3120_amd1_idl_


#include <itut_m3120.idl>


#pragma prefix "itu.int"


/**
Este código IDL está destinado a su almacenamiento en un fichero denominado
"itut_m3120_amd1.idl_" situado en el trayecto de búsqueda utilizado por los
compiladores IDL en vuestro sistema. El módulo principal M.3120 (definido
en M.3120) está contenido en un fichero separado denominado " itut_m3120.idl".
*/



/**
```

```
Este fragmento se añade al módulo, itut_m3120, que contiene la definición IDL
basada en objetos definidos en M.3100 y G.855.1.
*/
module itut_m3120
{



/**
```

4.1 Importaciones

```
*/

/**
Tipos importados de itut_x780
*/
typedef itut_x780::AdditionalTextType AdditionalTextType;
typedef itut_x780::AdditionalInformationSetType AdditionalInformationSetType;
typedef itut_x780::CorrelatedNotificationSetType CorrelatedNotificationSetType;
typedef itut_x780::NotifIDType NotifIDType;
typedef itut_x780::NullType NullType;




/**
```

4.2 Declaraciones directas

```
*/

/**
Declaraciones directas de interfaz
*/
     interface ProtectionGroup;
     interface ProtectionUnit;



/**
Declaraciones directas de valuetype
*/
     valuetype ProtectionGroupValueType;
     valuetype ProtectionUnitValueType;


/**
Declaraciones directas de typedefs
*/
     typedef MONameType ProtectionGroupNameType;
     typedef MONameType ProtectionUnitNameType;




/**
```

4.3 Estructuras y Typedefs

```
Las estructuras y typedefs definidos a continuación se disponen en orden
alfanumérico.
*/

     typedef sequence<ProtectionGroupNameType> ProtectionGroupNameSetType;

     typedef sequence<ProtectionGroupNameType> ProtectionGroupNameSeqType;

     typedef sequence<ProtectionUnitNameType> ProtectionUnitNameSetType;

     typedef sequence<ProtectionUnitNameType> ProtectionUnitNameSeqType;

/**
Este atributo especifica los criterios de la condición de enganche. Los
criterios incluyen la velocidad de conmutación automática de protección (APS,
automatic protection switching) y la fijación y liberación de ventanas de
tiempo. Si el número de APS de una unidad de protección alcanza el valor
especificado en el campo hitsCount dentro de una ventana de tiempo en movimiento
de longitud especificada, la unidad de protección pasará a la condición de
enganche. La longitud de la ventana de tiempo para pasar a la condición de
enganche se especifica en el campo settingWindowTime. Una vez que una unidad de
protección está en la condición de enganche, se denegará toda futura petición de
APS hasta que se libere la condición de enganche. El criterio de liberación es
que no haya ninguna petición APS dentro de otra ventana de tiempo en movimiento.
La longitud de esta ventana de tiempo se especifica en el campo
releasingWindowTime.
*/
     struct LockedInConditionType
     {
          unsigned short settingWindowTime;
          unsigned short releaseingWindowTime;
          unsigned short hitsCount;
     };

     enum ProtectionDirectionType
     {
          protectionDirectionTransmit,
          protectionDirectionReceive,
          protectionDirectionBidirectional
     };

     enum ProtectionGroupType
     {
          protectionGroupPlus, // 1+1 or hot-standby
          protectionGroupColon // M:N
     };

/**
El atributo estado de protección indica el estado de la conmutación de
protección en un objeto unidad de protección. Tiene el siguiente comportamiento:

-    Debe ser capaz de indicar las peticiones de conmutación pendientes, así
     como las activas, relativas a la unidad de protección. No obstante, al
     mismo tiempo sólo puede estar presente uno de los valores desenganche,
     conmutación forzada o conmutación manual.

-    Un sistema de protección puede soportar únicamente un subconjunto de los
     posibles valores de este atributo. El subconjunto de valores que debe
     soportar un sistema depende de la implementación.

-    La sintaxis de este atributo incluye un subcampo "unidad conexa"
     ("relatedUnit") cuyo contenido es la unión de "fromProtectionUnitNumber" y
```

"toProtectionUnitNumber". Este subcampo se utiliza para indicar en qué unidad se cursa el servicio.

Para una PU protegida, tanto fromProtectionUnit como toProtectionUnit conservan la ID de la PU protectora correspondiente. Al conmutar a la PU protectora (es decir que el servicio correrá a cargo de la PU protectora), se utiliza toProtectionUnit. Al volver a la PU protegida (el servicio correrá a cargo de la PU protegida) se utiliza fromProtectionUnit.

Para una PU protectora, tanto fromProtectionUnit como toProtectionUnit conservan la ID de la PU protegida correspondiente. Al conmutar a la PU protegida (es decir que el servicio correrá a cargo de la PU protegida), se utiliza toProtectionUnit. Al volver a la PU protectora (el servicio correrá a cargo de la PU protectora) se utiliza fromProtectionUnit.

Si un sistema puede soportar la conmutación de protección por degradación de recurso (RD, *resource degrade*) además de fallo de recurso (RF, *resource fail*), la conmutación de protección RD es similar a la que se indica en la descripción subsiguiente para RF.

Los siguientes valores admisibles de estado de protección están asociados a cada unidad de protección (PU) protegida:

- No hay petición: No hay en la unidad ninguna petición de conmutación. En este caso, el servicio se cursa en la PU protegida, la sintaxis de estado es noRequest. Para un sistema no reversible, la sintaxis de estado de la PU protectora correspondiente es también noRequest.

- Conmutación manual a la unidad protectora completa: La unidad ha completado una conmutación manual. En este caso, el servicio se cursa en la protectingPU correspondiente, la sintaxis de estado de la PU protegida es manualSwitch (switchStatus: completed; relatedUnit: toPU). La sintaxis de estado de la PU protectora correspondiente es manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Fallo de liberación: Ocurre una liberación mientras se espera una liberación. En este caso, el servicio está todavía en la protectingPU, la sintaxis de estado es releaseFailed más el estado anterior, por ejemplo manualSwitch (switchStatus: completed; relatedUnit: toPU). La sintaxis de estado de la PU protectora correspondiente está todavía en el estado anterior, por ejemplo manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Conmutación automática (RF) pendiente: La unidad tiene una condición de fallo presente y la unidad protectora está indisponible. En este caso, el servicio está todavía en la protectedPU, la sintaxis de estado es autoSwitch (switchStatus: pending; relatedUnit: toPU; reason: RF). La sintaxis de estado de la PU protectora correspondiente es autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF) más su estado previo.

- Conmutación automática (RF) completa: La unidad ha completado una conmutación automática a la unidad protectora debido a una condición de fallo de equipo. En este caso, el servicio está en la PU protectora correspondiente, la sintaxis de estado de la PU protegida es autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: RF). La sintaxis de estado de la PU protectora correspondiente es autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF.)

- Conmutación automática (RF) presente, fallo de operación: Está en curso una petición de conmutación automática (RF) y ocurre una expiración mientras se espera la compleción. En este caso, el servicio está todavía en la protectedPU, la sintaxis de estado es autoSwitch (switchStatus: failed; relatedUnit: toPU; reason: RF). La sintaxis de estado de la PU protectora

correspondiente es autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF) más su estado anterior.

- Conmutación forzada completa, conmutación automática (RF) pendiente: La unidad ha completado una conmutación forzada. Además, tiene pendiente una conmutación automática (RF). En este caso, el servicio está en la protectingPU correspondiente, la sintaxis de estado de la PU protegida es forceSwitch (switchStatus: completed; relatedUnit: toPU) más autoSwitch (switchStatus: pending; relatedUnit: toPU; reason: RF). La sintaxis de estado de la PU protectora correspondiente es forceSwitch (switchStatus: completed; relatedUnit: fromPU) más autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF).

- Conmutación automática completa, en espera de restablecimiento (únicamente en el caso reversible): La unidad ha completado una conmutación automática a la unidad protectora. En este caso, el servicio está en la protectingPU correspondiente, la sintaxis de estado de la PU protegida es autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: WTR). La sintaxis de estado de la PU protectora correspondiente es autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: WTR).

- Conmutación forzada completa: La unidad ha completado una conmutación forzada a la unidad protectora. En este caso, el servicio está en la protectingPU correspondiente, la sintaxis de estado de la PU protegida es forceSwitch (switchStatus: completed; relatedUnit: toPU). La sintaxis de estado de la PU protectora correspondiente es forceSwitch (switchStatus: completed; relatedUnit: fromPU).

- Desenganche de la unidad protegida completada: La unidad ha sido desenganchada de la unidad protectora. En este caso, el servicio está en la protectedPU, la sintaxis de estado es lockout (switchStatus: completed).

- Desenganche de la unidad protegida, fallo de operación: La unidad ha sido desenganchada de la unidad protectora y la conmutación anteriormente completada no ha podido ser liberada dentro del tiempo de expiración previsto. Cuando la conmutación es liberada, se retira el estado fallo de operación. En este caso, el servicio está todavía en la protectingPU correspondiente, la sintaxis de estado de la PU protegida es desenganche (switchStatus: completed) más releaseFailed. La sintaxis de estado de la PU protectora correspondiente es aún el estado anterior, por ejemplo manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Enganche: La unidad está en la condición de enganchada. Esto es causado por una cantidad excesiva de eventos de conmutación de protección. En este caso, el servicio está en la protectedPU, la sintaxis de estado es locked-in.

Una unidad de protección protegida no reversible tiene los siguientes valores de estado adicionales:

- No reversión: La unidad protegida ha sido conmutada a la unidad protectora y la petición de hacerlo ha sido liberada. La conmutación a la unidad protectora se mantiene. En este caso, el servicio está en la protectingPU correspondiente, la sintaxis de estado de la PU protegida es doNotRevert. La sintaxis de estado de la PU protectora correspondiente es doNotRevert.

- Conmutación manual a la unidad protegida completa: La unidad ha completado una conmutación manual de la unidad protectora a la unidad protegida. En este caso, el servicio está en la protectedPU, la sintaxis de estado es manualSwitch (switchStatus: completed; relatedUnit: fromPU). La sintaxis de estado de la PU protectora correspondiente es manualSwitch (switchStatus: completed; relatedUnit: toPU).

- Conmutación forzada a la unidad protegida completa: La unidad ha completado una conmutación forzada de la unidad protectora a la unidad protegida. En este caso, el servicio está en la protectedPU, la sintaxis de estado es forceSwitch (switchStatus: completed; relatedUnit: fromPU). La sintaxis de estado de la PU protectora correspondiente es forceSwitch (switchStatus: completed; relatedUnit: toPU).

- Conmutación automática (RF) a la unidad protegida completa: En la unidad protectora hay una condición fallo de equipo, y está utilizándose ahora la unidad protegida. En este caso, el servicio está en la protectedPU, la sintaxis de estado es autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF). La sintaxis de estado de la PU protectora correspondiente es autoSwitch (switchStatus: completed; relatedUnit: toPU; reason; RF).

- Conmutación forzada desde la unidad protectora completa, conmutación automática (RF) pendiente: La unidad ha completado una conmutación forzada de la unidad protectora a la unidad protegida. Además, en la unidad protegida hay una condición conmutación automática (RF). En este caso, el servicio está en la protectedPU, la sintaxis de estado es forceSwitch (switchStatus: completed; relatedUnit: fromPU) más autoSwitch (switchStatus: pending; relatedUnit: toPU; reason: RF). La sintaxis de estado de la PU protectora correspondiente es forceSwitch (switchStatus: completed; relatedUnit: toPU) más autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF).

Cada unidad de protección protectora tiene asociados los siguientes valores admisibles de estado de protección:

- No hay petición: No está presente ninguna petición de conmutación en la unidad protectora. En este caso, el servicio no está en la PU protectora, la sintaxis de estado es noRequest. Para los sistemas no reversibles, la sintaxis de estado de la PU protegida correspondiente es noRequest.

- Conmutación manual a la unidad protectora completa: Una unidad protegida ha completado una conmutación manual. En este caso, el servicio está en la protectingPU, la sintaxis de estado es manualSwitch (switchStatus: completed; relatedUnit: fromPU). La sintaxis de estado de la PU protegida correspondiente es manualSwitch (switchStatus: completed; relatedUnit: toPU).

- Conmutación automática (RF) pendiente: Hay una condición fallo de equipo en una unidad protegida, y la unidad protectora está indisponible para esta petición. En este caso, el servicio permanece en la protectedPU. La sintaxis de estado de la protectedPU es autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason RF) más su estado anterior, lo que causa su indisponibilidad. La sintaxis de estado de la PU protegida correspondiente es autoSwitch (switchStatus: pending; relatedUnit: toPU; reason: RF).

- Conmutación automática (RF) a la unidad protectora completa: Una unidad protegida ha completado una conmutación automática (RF) a la unidad protectora. En este caso, el servicio está en la protectingPU, la sintaxis de estado es autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF). La sintaxis de estado de la PU protegida correspondiente es autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: RF).

- Conmutación automática (RF) a la unidad protectora completa, en espera de restablecimiento (únicamente en el caso reversible): La unidad ha completado una conmutación automática a partir de la unidad protegida. En este caso, el servicio está en la protectingPU, la sintaxis de estado es autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: WTR). La sintaxis de estado de la PU protegida correspondiente es autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: WTR).

- RF presente en la unidad protectora: En la unidad protectora hay una condición de fallo de equipo. La sintaxis de estado de la PU protectora es resourceFailed.

- Conmutación forzada a la unidad protectora completa: La unidad ha completado una conmutación forzada de una unidad protegida a la unidad protectora. En este caso, el servicio está en la protectingPU, la sintaxis de estado es forceSwitch (switchStatus: completed; relatedUnit: fromPU). La sintaxis de estado de la PU protegida correspondiente es forceSwitch (switchStatus: completed; relatedUnit: toPU).

- Unidad protectora desenganchada: La unidad protectora ha sido desenganchada. En este caso, el servicio no está en la protectingPU, la sintaxis de estado es desenganche (switchStatus: completed).

- Fallo de liberación de desenganche de la unidad protectora: Al estar en curso una liberación de desenganche expira un temporizador de espera de liberación de la condición de desenganche. En este caso, el servicio no está en la protectingPU, la sintaxis de estado es lockout (switchStatus: failed).

Una unidad de protección protectora no reversible tiene los siguientes valores de estado adicionales:

- No reversión: Una unidad protegida ha sido conmutada a la unidad protectora y la petición de hacerlo ha sido liberada. Se mantiene la conmutación a la unidad protectora. En este caso, el servicio está en la protectingPU, la sintaxis de estado es doNotRevert. La sintaxis de estado de la PU protegida correspondiente es doNotRevert.

- Conmutación manual a la unidad protegida completa: La unidad ha completado una conmutación manual de la unidad protectora a la unidad protegida. En este caso, el servicio está en la protectedPU. La sintaxis de estado de la PU protegida es manualSwitch (switchStatus: completed; relatedUnit: toPU). La sintaxis de estado de la PU protegida correspondiente es manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Conmutación forzada a la unidad protegida completa: La unidad protectora ha completado una conmutación forzada a la unidad protegida. En este caso, el servicio está en la protectedPU. La sintaxis de estado de la PU protectora es forceSwitch (switchStatus: completed; relatedUnit: toPU). La sintaxis de estado de la PU protegida correspondiente es forceSwitch (switchStatus: completed; relatedUnit: fromPU).

- Conmutación forzada a la unidad protegida completa, fallo de equipo de la unidad protectora: La unidad protectora ha completado una conmutación forzada a la unidad protegida. Además, hay una condición fallo de equipo en la unidad protectora. En este caso, el servicio está en la protectedPU. La sintaxis de estado de la PU protectora es forceSwitch (switchStatus: completed; relatedUnit: toPU) más equipmentFailed. La sintaxis de estado de la PU protegida correspondiente es forceSwitch (switchStatus: completed; relatedUnit: fromPU).

- Conmutación automática (RF) a la unidad protegida completa: Hay una condición de fallo de equipo en la unidad protectora, y está utilizándose la unidad protegida. En este caso, el servicio está en la protectedPU. La sintaxis de estado de la PU protectora es autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: RF). La sintaxis de estado de la PU protegida correspondiente es autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF).

*/

```
    enum PAutoSwitchReasonType
    {
```

```
     pAutoSwitchReasonWaitToRestore,
     pAutoSwitchReasonSignalDegrade,
     pAutoSwitchReasonSignalFail
};

enum PRelatedUnitChoiceType
{
     pFrom,
     pTo
};

enum PSwitchStatusType
{
     pSwitchStatusPending,
     pSwitchStatusCompleted,
     pSwitchStatusFailed
};

enum PLockoutChoiceType
{
     pSwitchStatus,
     pReleaseFailed
};

union PRelatedUnitType switch (PRelatedUnitChoiceType)
{
     case pFrom:
        ProtectionUnitNameType fromUnit;
     case pTo:
        ProtectionUnitNameType toUnit;
};

struct PManualOrForcedSwitchType
{
     PSwitchStatusType      switchStatus;
     PRelatedUnitType       relatedUnit;
};

struct PAutoSwitchType
{
     PSwitchStatusType      switchStatus;
     PRelatedUnitType       relatedUnit;
     PAutoSwitchReasonType  autoSwitchReason;
};

union PLockoutType switch (PLockoutChoiceType)
{
     case pSwitchStatus:
        PSwitchStatusType   switchStatus;
     case pReleaseFailed:
        NullType            releaseFailed;

};

enum ProtectionStatusChoiceType
{
     protectionStatusNoRequest,
     protectionStatusDoNotRevert,
     protectionStatusManualSwitch,
     protectionStatusAutoSwitch,
     protectionStatusForcedSwitch,
     protectionStatusLockout,
     protectionStatusReleaseFailed,
     protectionStatusResourceFailed,
```

```
        protectionStatusLockedIn
    };

    union ProtectionStatusType switch (ProtectionStatusChoiceType)
    {
        case protectionStatusNoRequest:
            NullType      noRequest;
        case protectionStatusDoNotRevert:
            NullType      doNotRevert;
        case protectionStatusManualSwitch:
            PManualOrForcedSwitchType   manualSwitch;
        case protectionStatusAutoSwitch:
           PAutoSwitchType              autoSwitch;
        case protectionStatusForcedSwitch:
           PManualOrForcedSwitchType    forcedSwitch;
        case protectionStatusLockout:
           PLockoutType                 lockout;
        case protectionStatusReleaseFailed:
            NullType      releaseFailed;
        case protectionStatusResourceFailed:
            NullType      resourceFailed;
        case protectionStatusLockedIn:
            NullType      lockedIn;
    };

    struct ProtectionStatusType
    {
        ProtectionStatusChoiceType      choice;
        PSwitchStatusType               switchStatus;
           // may be NULL
        PRelatedUnitType                relatedUnit;
           // may be NULL
        PAutoSwitchReasonType           autoSwitchReason;
           // may be NULL
    };

    typedef sequence<ProtectionStatusType> ProtectionStatusSetType;

    enum PSwitchActionResultType
    {
        pSwitchActionResultPreempted,
        pSwitchActionResultFailure,
        pSwitchActionResultTimeout
    };

    enum PSwitchType
    {
        pSwitchManual,
        pSwitchForced,
        pSwitchLocked
    };



/**
```

4.4 Excepciones

```
*/

/**
```

4.4.1 Excepciones y constantes para lotes condicionales

```
*/

     exception NOpriorityPackage {};

     exception NOprotectionAlarmPackage {};

/**
Este conjunto de constantes representan los lotes condicionales definidos en
esta Recomendación. Cuando un lote condicional es soportado en un ejemplar de
objeto gestionado determinado, una de estas cadenas estará contenida en el
atributo lotes.
*/
       const string priorityPackage =
           "itut_m3120::priorityPackage";
       const string protectionAlarmPackage =
          "itut_m3120::protectionAlarmPackage";




/**
```

4.5 Interfaces – Granularidad fina

```
*/

/**
```

4.5.1 Grupo de protección

Esta clase de objeto gestionado se utiliza para representar un sistema de protección en un elemento de red (NE, *network element*). Las funciones de gestión primarias de esta entidad son las notificaciones de eventos de conmutación de protección y el control del sistema de gestión de desenganches, conmutaciones forzadas y conmutaciones manuales.

Pueden crearse automáticamente ejemplares de esta clase de objeto en un agente, por ejemplo, siguiendo inmediatamente a la inicialización de los recursos NE que intervienen en el sistema de protección, de conformidad con la sustitución y el modo del NE. En el agente pueden suprimirse automáticamente los ejemplares de esta clase de objeto.

En un NE pueden existir varios ejemplares del objeto protectionGroupR2 (una para cada sistema de protección soportado por el NE). Un ejemplar del objeto ProtectionGroup puede contener dos o más ejemplares del objeto gestionado ProtectionUnit.

O todos o ninguno de los lotes unidad de protección de un objeto grupo de protección tendrá el lote priorityPkg. Es de señalar que, antes de la creación del objeto ProtectionGroup, el atributo soportado por lista de objetos (sbol) de un recurso fiable tal como un punto de terminación puede apuntar a un objeto recurso no fiable tal como paquete de circuitos. Pero una vez creado el objeto el atributo sbol comenzaría a apuntar al objeto grupo de protección.

Esta clase de objeto tiene los siguientes atributos:

– Estado operacional: Este atributo de sólo lectura indica que el mecanismo de protección representado por este ejemplar es capaz, o no, de realizar sus funciones normales.

– Tipo de grupo de protección: Este atributo de lectura-escritura indica que el esquema de protección utilizado es 1+1, o M:N.

– Reversible: Este atributo de lectura-escritura indica que el esquema de protección utilizado es reversible, o no. El valor por defecto de este atributo indicará funcionamiento reversible, pero, mediante una instrucción del administrador, este atributo debe poder indicar funcionamiento no reversible.

– Tiempo de espera hasta el restablecimiento: Este atributo de lectura-escritura identifica la cantidad de tiempo, en segundos, que el sistema de protección debe esperar después de la solución de un fallo antes de volver al recurso protegido. Este atributo es relevante únicamente para el funcionamiento reversible.

– Condición de enganche: Este atributo de lectura-escritura especifica los criterios de la condición de enganche.

Esta interfaz contiene los siguientes LOTES CONDICIONALES.

– protectionAlarmPackage: Presente si el sistema es capaz de comunicar fallo del mecanismo de protección o fallo del recurso protector.

– createDeleteNotificationsPackage: Presente si un ejemplar lo soporta.

– attributeValueChangeNotificationPackage: Presente si un ejemplar lo soporta.

*/

```
valuetype ProtectionGroupValueType: itut_x780::ManagedObjectValueType
{
     public OperationalStateType      operationalState;
        // GET
     public AvailabilityStatusSetType availabilityStatus;
        // GET
     public MONameSetType             supportedByObjectList;
        // GET-REPLACE, ADD-REMOVE
     public ProtectionGroupType       groupType;
        // GET-REPLACE
     public boolean                   revertive;
        // GET-REPLACE-WITH-DEFAULT
     public unsigned short            waitToRestoreTime;
        // GET-REPLACE
     public LockedInConditionType     lockedInCondition;
        // GET-REPLACE
     public CurrentProblemSetType     currentProblemList;
        // conditional
        // protectionAlarmPackage
        // GET
}; // valuetype ProtectionGroupValueType


interface ProtectionGroup: itut_x780::ManagedObject
{
     const boolean revertiveDefault = TRUE;

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

```
        AvailabilityStatusSetType availabilityStatusGet ()
           raises (itut_x780::ApplicationError);


        MONameSetType supportedByObjectListGet ()
           raises (itut_x780::ApplicationError);

        void supportedByObjectListSet
           (in MONameSetType objectList)
           raises (itut_x780::ApplicationError);

        void supportedByObjectListAdd
           (in MONameSetType objectList)
           raises (itut_x780::ApplicationError);

        void supportedByObjectListRemove
           (in MONameSetType objectList)
           raises (itut_x780::ApplicationError);


        ProtectionGroupType groupTypeGet ()
           raises (itut_x780::ApplicationError);


        boolean revertiveGet ()
           raises (itut_x780::ApplicationError);

        void revertiveSet
           (in boolean revertive)
           raises (itut_x780::ApplicationError);


        unsigned short waitToRestoreTimeGet ()
           raises (itut_x780::ApplicationError);

        void waitToRestoreTimeSet
           (in unsigned short waitTime)
           raises (itut_x780::ApplicationError);


        LockedInConditionType lockedInConditionGet ()
           raises (itut_x780::ApplicationError);

        void lockedInConditionSet
           (in LockedInConditionType condition)
           raises (itut_x780::ApplicationError);


        CurrentProblemSetType currentProblemListGet ()
           raises (itut_x780::ApplicationError,
                     NOprotectionAlarmPackage);
```

/**
Invocación de protección: Esta acción se utiliza para solicitar un desenganche, una conmutación forzada o una conmutación manual en uno o varios ejemplares ProtectionUnit contenidos en el objeto ProtectionGroup. En la acción invocación de protección se incluyen los siguientes parámetros de entrada:

– Tipo de conmutación (manual, forzada o desenganche).

– Entidad de protección (opcional): ID de la entidad unidad de protección protegida y/o protectora a la que se aplica la petición.

```
Si no está presente, se supone que la petición se aplica a todas las entidades
de ese tipo en el grupo de protección.

Si una unidad protectora está identificada en el campo protectedUnits, o si una
unidad protectora está identificada en el campo protectingUnits, la acción
falla.

Si la petición es conmutación forzada o conmutación manual, el campo
protectedUnits identificará una o más unidades de protección. Si sólo una unidad
está identificada en el campo protectedUnits, y hay solamente una unidad
protectora en el grupo de protección, puede omitirse el campo protectingUnits.
Si el campo protectingUnits está presente, identificará el mismo número de
unidades que el campo protectedUnits.

Si la petición es desenganche, el campo entidad de protección puede estar
ausente, indicando que la petición se aplica a todas las unidades de protección
contenidas. Si el campo entidad de protección está presente, cualquier número de
unidades de protección pueden ser identificadas en el campo protectedUnits y/o
protectingUnits, y puede estar ausente uno u otro campo.

Para una petición desenganche, se desenganchan las unidades protegidas y/o
unidades protectoras especificadas.

Para peticiones que no pueden ser completadas, ya sea porque la petición en la
unidad protectora está atendiendo una petición de prioridad superior, o porque
se produce fallo, o se produce temporización (timeout), la respuesta indicará
por qué no pudo completarse la petición, y la petición no se hará pendiente.

*/
      void invokeProtection
            (in PSwitchType switchType,
            in ProtectionUnitNameSeqType protectedUnitList,
            in ProtectionUnitNameSeqType protectingUnitList,
            out PSwitchActionResultType actionResult)
            raises (itut_x780::ApplicationError);

/**
Liberación de protección: Esta acción se utiliza para liberar un desenganche,
una conmutación forzada o una conmutación manual en uno o varios ejemplares
protectionUnit contenidos en el objeto protectionGroup. En la acción liberación
de protección se incluyen los siguientes parámetros de entrada:

–    Tipo de conmutación (manual, forzada o desenganche).

–    Entidad de Protección (opcional): ID(s) de la entidad unidad de protección
     protegida y/o protectora a la que se aplica la petición.

Si no está presente, se supone que la petición se aplica a todas las entidades
de ese tipo en el grupo de protección.

Si una unidad protectora está identificada en el campo protectedUnits, o si una
unidad protectora está identificada en el campo protectingUnits, la acción
falla.

Si la petición es conmutación forzada o conmutación manual, el campo
protectedUnits identificará una o más unidades de protección, y se omitirá el
campo protectingUnits. Para cada unidad protegida identificada, si no está
conmutada a una unidad de protección, la acción falla.

Si la petición es desenganche, el campo protectionEntity puede estar ausente,
indicando que la petición se aplica a todas las unidades de protección
contenidas. Si el campo entidad de protección está presente, cualquier número de
unidades de protección pueden ser identificadas en el campo protectedUnits y/o
protectingUnits, y puede estar ausente uno u otro campo.
```

```
Para una petición desenganche, las unidades protegidas y/o unidades protectoras
especificadas ya no son desenganchadas. Es decir, las unidades protegidas están
ahora bajo protección y las unidades protectoras no son capaces de proporcionar
protección.

Para peticiones de liberación que no puedan ser completadas, la respuesta
indicará por qué la petición no pudo ser completada.

*/
       void releaseProtection
          (in PSwitchType switchType,
          in ProtectionUnitNameSeqType protectedUnitList,
          in ProtectionUnitNameSeqType protectingUnitList,
          out PSwitchActionResultType actionResult)
          raises (itut_x780::ApplicationError);


       MANDATORY_NOTIFICATION(
              Notifications, protectionSwitchReporting)
       MANDATORY_NOTIFICATION(
              itut_x780::Notifications, stateChange)

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

    };  // interface ProtectionGroup


    interface ProtectionGroupFactory: itut_x780::ManagedObjectFactory
    {
      itut_x780::ManagedObject create
         (in NameBindingType nameBinding,
         in MONameType superior,
         in string reqID,   // auto naming if empty string
         out MONameType name,
         in StringSetType packageNameList,
         in MONameSetType supportedByObjectList,
          // GET-REPLACE, ADD-REMOVE
         in ProtectionGroupType groupType,
          // GET-REPLACE
         in boolean revertive,
          // GET-REPLACE-WITH-DEFAULT
         in unsigned short waitToRestoreTime,
          // GET-REPLACE
         in LockedInConditionType lockedInCondition)
          // GET-REPLACE
         raises (itut_x780::ApplicationError,
          itut_x780::CreateError);

    }; // interface ProtectionGroupFactory


/**
```

4.5.2 Unidad de protección

La clase de objeto gestionado ProtectionUnit se utiliza para representar el recurso protegido (es decir, el que está funcionando o funciona regular o preferentemente) o proteger (es decir, de reserva o en espera) en un sistema de protección. Relaciona los recursos (por ejemplo, los conjuntos de circuitos) que intervienen en el sistema de protección y lleva la reseña del estado de conmutación de protección de los recursos.

Los ejemplares de esta clase de objeto son creados automáticamente por el agente con arreglo a los esquemas de protección adoptados por el NE. Un ejemplar unidad de protección es suprimido cuando el ejemplar objeto de recurso apuntada por el atributo lista de recursos no fiables es suprimido, y puede ser creado automáticamente cuando se crea el objeto de recurso asociado. El agente puede también crear y suprimir ejemplares de esta clase de objeto con el fin de reflejar las modificaciones locales en los esquemas de protección. La notificación attributeValueChange se utiliza para notificar cambios de los atributos lista de recursos fiables, estado de protección y prioridad.

En un ejemplar del objeto ProtectionGroup pueden existir dos o más ejemplares del objeto ProtectionUnit.

Un ejemplar del objeto ProtectionUnit podría contener un ejemplar del objeto ProtectionCurrentData.

Un ejemplar ProtectionUnit está relacionado con ejemplares de entidades de recursos (por ejemplo, conjuntos de circuitos) a través del atributo lista de recursos no fiables. Si la función de las entidades de recurso (por ejemplo, función de temporización, función de punto de terminación de transporte, etc.) está modelada explícitamente como ejemplares de objeto en el NE, un ejemplar ProtectionUnit también está relacionado con ejemplares de la entidad de función modelada a través del atributo lista de recursos fiables.

Esta clase de objeto tiene los atributos siguientes:

- Protectora: Este atributo de sólo lectura indica que la unidad de protección está asociada a un recurso que le da un papel de protectora ("verdadero") o de protegida ("falso") en el sistema de protección.

- Lista de recursos no fiables: Este atributo de sólo lectura identifica el recurso no fiable (por ejemplo, entidad conjunto de circuitos) asociado al objeto Unidad de Protección (por ejemplo, el recurso protegido o protector). La sintaxis de este atributo incluye un conjunto de valores y puede apuntar a múltiples ejemplares de recurso no fiable cuando un conjunto de recursos forma una unidad atómica en el sistema de protección.

- Lista de recursos fiables: Este atributo de sólo lectura identifica el recurso fiable (es decir, la entidad funcional), si lo hay, asociado a la Unidad de Protección. El valor de este atributo de una unidad de protección (PU, *protection unit*) cambiará cuando la PU intervenga en una conmutación o liberación de protección. Para una PU protegida, cuando no está conmutada, este atributo apunta al recurso fiable asociado (es decir, el objeto funcional); cuando está conmutada, este atributo apunta a nulo (NULL). Para una PU protectora, cuando no está conmutada, este atributo apunta a NULL; cuando está conmutada, este atributo apunta al recurso fiable asociado (es decir, el objeto funcional). La sintaxis de este atributo incluye un conjunto de valores y puede apuntar a múltiples ejemplares de recursos fiables cuando un conjunto de objetos funcionales forma una unidad atómica en el sistema de protección.

- Prioridad: Este atributo de lectura-escritura especifica la prioridad del servicio transportado en el recurso asociado al ejemplar de unidad de protección. Los valores admitidos de este atributo son enteros; el valor 1 indica la prioridad más alta, y un valor más grande indica una prioridad más baja.

- Estado de protección R1: Este atributo de sólo lectura indica el estado de la conmutación de protección en un objeto unidad de protección. Tiene el siguiente comportamiento:

  - Debe ser capaz de indicar las peticiones de conmutación pendientes, así como las activas, relativas a la unidad de protección. No obstante, al mismo tiempo sólo puede estar presente uno de los valores desenganche, conmutación forzada o conmutación manual.

  - Un sistema de protección puede soportar únicamente un subconjunto de los posibles valores de este atributo. El subconjunto de valores que debe soportar un sistema depende de la implementación.

Esta interfaz contiene los siguientes LOTES CONDICIONALES.

- priorityPackage: Presente si un ejemplar lo soporta.

- attributeValueChangeNotificationPackage: Presente si un ejemplar lo soporta.

```
*/
     valuetype ProtectionUnitValueType: itut_x780::ManagedObjectValueType
     {
          public boolean                  protecting;
              // GET
          public MONameSetType            reliableResourceList;
              // GET
          public MONameSetType            unreliableResourceList;
              // GET
          public ProtectionStatusSetType  protectionStatus;
              // GET
          public unsigned short           priority;
              // conditional
              // priorityPackage
              // GET-REPLACE
     }; // valuetype ProtectionUnitValueType


     interface ProtectionUnit : itut_x780::ManagedObject
     {
       boolean protectingGet ()
          raises (itut_x780::ApplicationError);

/**

El valor del atributo reliableResourceList apunta al recurso (o recursos) fiable
(por ejemplo, los objetos funcionales) que está o están asociados con el
ejemplar unidad de protección.
*/
       MONameSetType reliableResourceListGet ()
          raises (itut_x780::ApplicationError);

/**
El valor del atributo unreliableResourceList apunta al recurso (o recursos) no
fiable (por ejemplo, los conjuntos de circuitos) que está o están asociados con
el ejemplar unidad de protección.
*/
       MONameSetType unreliableResourceListGet ()
          raises (itut_x780::ApplicationError);
```

```
/**
*/
       ProtectionStatusSetType protectionStatusGet ()
          raises (itut_x780::ApplicationError);

/**
Este atributo especifica la prioridad del servicio (por ejemplo tráfico)
transportado en el recurso asociado al ejemplar ProtectionUnit. Los valores
admitidos de este atributo son enteros; el valor 1 indica la prioridad más alta,
y un valor más grande indica una prioridad más baja.

Para una ProtectionUnit protectora, el valor de este atributo indica la
prioridad de elección de la ProtectionUnit protectora con relación a otra u
otras ProtectionUnit protectoras disponibles dentro del mismo ProtectionGroup. A
menor valor, más preferida es la ProtectionUnit con relación a las otras
ProtectionUnits.

*/
       unsigned short priorityGet ()
          raises (itut_x780::ApplicationError,
               NOpriorityPackage);

       void prioritySet
          (in unsigned short priority)
          raises (itut_x780::ApplicationError,
               NOpriorityPackage);

       CONDITIONAL_NOTIFICATION(
          itut_x780::Notifications, attributeValueChange,
          attributeValueChangeNotificationPackage)

     }; // interface ProtectionUnit


     interface ProtectionUnitFactory: itut_x780::ManagedObjectFactory
     {
       itut_x780::ManagedObject create
          (in NameBindingType nameBinding,
          in MONameType superior,
          in string reqID,   // auto naming if empty string
          out MONameType name,
          in StringSetType packageNameList,
          in unsigned short priority)
              // conditional
              // priorityPackage
              // GET-REPLACE
          raises (itut_x780::ApplicationError,
               itut_x780::CreateError);

     }; // interface ProtectionUnitFactory


/**
```

4.6 Interfaces – Fachada

El comportamiento de las interfaces de fachada es idéntico al de las correspondientes interfaces de granularidad fina. Por tanto, no se incluyen comentarios en las interfaces de fachada. Se remite a los lectores a la interfaz de granularidad fina de 4.5 en relación con el comportamiento de la interfaz de fachada.

```
Esta cláusula puede omitirse del IDL si un sistema de gestión sólo soporta
interfaces de granularidad fina.
*/

/**
```

4.6.1 ProtectionGroup_F

```
*/

     interface ProtectionGroup_F: itut_x780::ManagedObject_F
     {
          const boolean revertiveDefault = TRUE;

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

          AvailabilityStatusSetType availabilityStatusGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);


          MONameSetType supportedByObjectListGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);

          void supportedByObjectListSet
              (in MONameType name,
              in MONameSetType objectList)
              raises (itut_x780::ApplicationError);

          void supportedByObjectListAdd
              (in MONameType name,
              in MONameSetType objectList)
              raises (itut_x780::ApplicationError);

          void supportedByObjectListRemove
              (in MONameType name,
              in MONameSetType objectList)
              raises (itut_x780::ApplicationError);


          ProtectionGroupType groupTypeGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);


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

          void revertiveSet
              (in MONameType name,
              in boolean revertive)
              raises (itut_x780::ApplicationError);


          unsigned short waitToRestoreTimeGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);

          void waitToRestoreTimeSet
              (in MONameType name,
              in unsigned short waitTime)
```

```
        raises (itut_x780::ApplicationError);


    LockedInConditionType lockedInConditionGet
        (in MONameType name)
        raises (itut_x780::ApplicationError);

    void lockedInConditionSet
        (in MONameType name,
        in LockedInConditionType condition)
        raises (itut_x780::ApplicationError);


    CurrentProblemSetType currentProblemListGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOprotectionAlarmPackage);


    void invokeProtection
        (in MONameType name,
        in PSwitchType switchType,
        in ProtectionUnitNameSeqType protectedUnitList,
        in ProtectionUnitNameSeqType protectingUnitList,
        out PSwitchActionResultType actionResult)
        raises (itut_x780::ApplicationError);

    void releaseProtection
        (in MONameType name,
        in PSwitchType switchType,
        in ProtectionUnitNameSeqType protectedUnitList,
        in ProtectionUnitNameSeqType protectingUnitList,
        out PSwitchActionResultType actionResult)
        raises (itut_x780::ApplicationError);


    MANDATORY_NOTIFICATION(
        Notifications, protectionSwitchReporting)
    MANDATORY_NOTIFICATION(
        itut_x780::Notifications, stateChange)

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

    };  // interface ProtectionGroup_F



/**
```

4.6.2 ProtectionUnit_F

```
*/

     interface ProtectionUnit_F : itut_x780::ManagedObject_F
     {
          boolean protectingGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);

          MONameSetType reliableResourceListGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);

          MONameSetType unreliableResourceListGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);

          ProtectionStatusType protectionStatusGet
              (in MONameType name)
              raises (itut_x780::ApplicationError);

          unsigned short priorityGet
              (in MONameType name)
              raises (itut_x780::ApplicationError,
              NOpriorityPackage);

          void prioritySet
              (in MONameType name,
              in unsigned short priority)
              raises (itut_x780::ApplicationError,
              NOpriorityPackage);

     }; // interface ProtectionUnit_F


/**
```

4.7 Notificaciones

```
*/
/**
Esta interfaz contiene las definiciones de las notificaciones emitidas por los
objetos gestionados.

Los usuarios de notificación que deseen utilizar notificaciones tipificadas
necesitan agregar las operaciones siguientes a la interfaz de notificación
M.3120 si la hubiere.

Los publicadores y abonados de notificaciones que deseen utilizar notificaciones
estructuradas basadas en las operaciones definidas a continuación deben seguir
las reglas de construcción y lectura de la estructura de notificación definida
en la Rec. UIT-T Q.816.

*/
          interface Notifications
     {

/**
La notificación protectionSwitchReporting es emitida desde el objeto
ProtectionGroup para comunicar cualesquiera eventos de conmutación de
protección.

*/
```

```
      void protectionSwitchReporting
            (in ExternalTimeType              eventTime,
            in MONameType                     source,
                  // protection group
            in ObjectClassType                sourceClass,
                  // always ProtectionGroup
            in NotifIDType                    notificationIdentifier,
            in CorrelatedNotificationSetType  correlatedNotifications,
            in AdditionalTextType             additionalText,
            in AdditionalInformationSetType   additionalInfo,
            in MONameType                     reportedProtectionUnit,
            in ProtectionStatusSetType        oldProtectionStatus,
            in ProtectionStatusSetType        newProtectionStatus,
            in ProtectionDirectionType        protectionDirection);

/**
La notificación protectionAlarm es emitida desde el objeto ProtectionGroup para
comunicar todo fallo del mecanismo de protección o fallo del recurso protector.

*/
      void protectionAlarm
            (in ExternalTimeType              eventTime,
            in MONameType                     source,
               // protection group
            in ObjectClassType                sourceClass,
               // always ProtectionGroup
            in NotifIDType                    notificationIdentifier,
            in CorrelatedNotificationSetType  correlatedNotifications,
            in AdditionalTextType             additionalText,
            in AdditionalInformationSetType   additionalInfo,
            in ProbableCauseType              probableCause);

    }; // interface Notfications

/**
```

4.8 Vinculación de nombres

```
*/

/**

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

4.8.1 ProtectionGroup

```
*/
      module ProtectionGroup_ManagedElement
      {
            const string   superiorClass =
             "itut_m3120::ManagedElement";
            const boolean  superiorSubclassesAllowed = TRUE;
            const string   subordinateClass =
                  "itut_m3120::ProtectionGroup";
            const boolean  subordinateSubclassesAllowed = TRUE;
            const boolean  managerCreatesAllowed = TRUE;
            const DeletePolicyType deletePolicy =
                  itut_x780::deleteOnlyIfNoContainedObjects;
            const string   kind = "ProtectionGroup";
      }; // module ProtectionGroup_ManagedElement
/**
```

4.8.2 ProtectionUnit

```
*/
       module ProtectionUnit_ProtectionGroup
       {
              const string   superiorClass =
                    "itut_m3120::ProtectionGroup";
               const boolean superiorSubclassesAllowed = TRUE;
              const string   subordinateClass =
                    "itut_m3120::ProtectionUnit";
              const boolean  subordinateSubclassesAllowed = TRUE;
              const boolean  managerCreatesAllowed = TRUE;
              const DeletePolicyType deletePolicy =
                    itut_x780::deleteOnlyIfNoContainedObjects;
              const string   kind = "ProtectionUnit";
       }; // module ProtectionUnit_ProtectionGroup


     }; // module NameBinding

/**
```

4.9 ProbableCauseConst

```
Este módulo contiene los valores constantes definidos del UID ProbableCause.
Los valores que siguen hay que agregarlos al módulo ProbableCauseConst
de M.3120.

*/

     module ProbableCauseConst
     {

/**
Los valores 81-100 están reservados para causas probables relacionadas con la
alarma del equipo.
*/
       const short protectionMechanismFailure = 81;
       const short protectingResourceFailure = 82;

     }; // module ProbableCauseConst




}; // module itut_m3120



#endif // _itut_m3120_amd1_idl_
```

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	Construcc ión, 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

Ginebra, 2002