



UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

X.722

Enmienda 3

(08/97)

SERIE X: REDES DE DATOS Y COMUNICACIÓN
ENTRE SISTEMAS ABIERTOS

Gestión de interconexión de sistemas abiertos –
Estructura de la información de gestión

Tecnología de la información – Interconexión de
sistemas abiertos – Estructura de la información de
gestión: Directrices para la definición de objetos
gestionados

**Enmienda 3: Directrices para la utilización de Z
en la formalización del comportamiento de
objetos gestionados**

Recomendación UIT-T X.722 – Enmienda 3

(Anteriormente Recomendación del CCITT)

RECOMENDACIONES DE LA SERIE X DEL UIT-T
REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

REDES PÚBLICAS DE DATOS	
Servicios y facilidades	X.1–X.19
Interfaces	X.20–X.49
Transmisión, señalización y conmutación	X.50–X.89
Aspectos de redes	X.90–X.149
Mantenimiento	X.150–X.179
Disposiciones administrativas	X.180–X.199
INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Modelo y notación	X.200–X.209
Definiciones de los servicios	X.210–X.219
Especificaciones de los protocolos en modo conexión	X.220–X.229
Especificaciones de los protocolos en modo sin conexión	X.230–X.239
Formularios para declaraciones de conformidad de implementación de protocolo	X.240–X.259
Identificación de protocolos	X.260–X.269
Protocolos de seguridad	X.270–X.279
Objetos gestionados de capa	X.280–X.289
Pruebas de conformidad	X.290–X.299
INTERFUNCIONAMIENTO ENTRE REDES	
Generalidades	X.300–X.349
Sistemas de transmisión de datos por satélite	X.350–X.399
SISTEMAS DE TRATAMIENTO DE MENSAJES	X.400–X.499
DIRECTORIO	X.500–X.599
GESTIÓN DE REDES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS Y ASPECTOS DE SISTEMAS	
Gestión de redes	X.600–X.629
Eficacia	X.630–X.639
Calidad de servicio	X.640–X.649
Denominación, direccionamiento y registro	X.650–X.679
Notación de sintaxis abstracta uno	X.680–X.699
GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Marco y arquitectura de la gestión de sistemas	X.700–X.709
Servicio y protocolo de comunicación de gestión	X.710–X.719
Estructura de la información de gestión	X.720–X.729
Funciones de gestión y funciones de arquitectura de gestión distribuida abierta	X.730–X.799
SEGURIDAD	X.800–X.849
APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Cometimiento, concurrencia y recuperación	X.850–X.859
Procesamiento de transacciones	X.860–X.879
Operaciones a distancia	X.880–X.899
PROCESAMIENTO DISTRIBUIDO ABIERTO	X.900–X.999

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

NORMA INTERNACIONAL 10165-4

RECOMENDACIÓN UIT-T X.722

TECNOLOGÍA DE LA INFORMACIÓN – INTERCONEXIÓN DE SISTEMAS ABIERTOS – ESTRUCTURA DE LA INFORMACIÓN DE GESTIÓN: DIRECTRICES PARA LA DEFINICIÓN DE OBJETOS GESTIONADOS

ENMIENDA 3

Directrices para la utilización de Z en la formalización del comportamiento de objetos gestionados

Resumen

Esta enmienda a la Rec. X.722 del CCITT | ISO/CEI 10165-4 contiene un ejemplo ilustrativo que muestra la mejor práctica actual en la utilización del lenguaje de descripción formal Z para especificar el comportamiento de objetos gestionados (MO, *managed object*). Tiene por objeto establecer una base y un entendimiento comunes para este planteamiento formal que ayudará a lograr coherencia en desarrollos similares. Proporciona un punto de partida útil para usuarios GDMO que deseen utilizar Z para mejorar sus especificaciones de comportamiento.

Las especificaciones formales del comportamiento de los objetos gestionados resultan valiosas ya que son claras y sin ambigüedades. El hecho de producir una especificación formal obliga a analizar detenidamente los detalles del comportamiento. Por lo tanto, también puede utilizarse como herramienta para identificar y corregir ambigüedades en una especificación que se mantendrá fundamentalmente en lenguaje natural.

Esta enmienda constituye una guía técnica sobre la utilización del lenguaje Z para la definición del comportamiento de objetos gestionados que soportan el interfuncionamiento de gestión OSI. Es informativo y no normativo. No requiere la utilización de técnicas de definición formal (FDT, *formal definition techniques*) para especificar el comportamiento de objetos gestionados. Si deben utilizarse FDT, no se requiere utilizar Z; también son adecuados otros lenguajes tales como el SDL.

Orígenes

El texto de la Recomendación UIT-T X.722, enmienda 3, se aprobó el 9 de agosto de 1997. Su texto se publica también, en forma idéntica, como Norma Internacional ISO/CEI 10165-4.

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 Conferencia Mundial de Normalización de las Telecomunicaciones (CMNT), 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 N.º 1 de la CMNT.

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 1998

Es propiedad. Ninguna parte de esta publicación puede reproducirse o utilizarse, de ninguna forma o por ningún medio, sea éste electrónico o mecánico, de fotocopia o de microfilm, sin previa autorización escrita por parte de la UIT.

ÍNDICE

	<i>Página</i>
1) Índice	1
2) Subcláusula 2.1	1
3) Nueva subcláusula 2.3.....	1
4) Nuevo anexo B.....	1
Anexo B – Directrices para la utilización de Z para la formalización del comportamiento de objetos gestionados.....	2

NORMA INTERNACIONAL

RECOMENDACIÓN UIT-T

**TECNOLOGÍA DE LA INFORMACIÓN – INTERCONEXIÓN DE
SISTEMAS ABIERTOS – ESTRUCTURA DE LA INFORMACIÓN
DE GESTIÓN: DIRECTRICES PARA LA DEFINICIÓN
DE OBJETOS GESTIONADOS**

ENMIENDA 3

**Directrices para la utilización de Z en la formalización del
comportamiento de objetos gestionados**

1) Índice

Añádase la siguiente referencia al índice:

Anexo B – Directrices para la utilización de Z en la formalización del comportamiento de objetos gestionados

2) Subcláusula 2.1

Añádase la siguiente referencia a 2.1:

- Recomendación X.731 del CCITT (1992) | ISO/CEI 10164-2:1992, Tecnología de la información – Interconexión de sistemas abiertos – Gestión de sistemas: Función de gestión de estados.

3) Nueva subcláusula 2.3

Añádase la nueva subcláusula como sigue:

2.3 Referencias adicionales

- ISO/CEI 13568:¹⁾, *Information technology – Z specification language*.

¹⁾ Actualmente en estado de proyecto.

4) Nuevo anexo B

Añádase el nuevo anexo B siguiente:

Anexo B

Directrices para la utilización de Z para la formalización del comportamiento de objetos gestionados

(Este anexo no es parte integrante de la presente Recomendación | Norma Internacional)

B.1 Introducción

Este anexo constituye una guía técnica sobre la utilización del lenguaje Z para la definición del comportamiento de objetos gestionados que soportan el interfuncionamiento de gestión OSI. Es informativo y no normativo. No requiere la utilización de técnicas de definición formal (FDT, *formal definition techniques*) para especificar el comportamiento de MO. Si deben utilizarse FDT, no se requiere utilizar Z; también son adecuados otros lenguajes como SDL. Incluso si ha de utilizarse Z, son posibles otras maneras de especificar el comportamiento de MO.

Pueden resultar directamente válidas especificaciones formales del comportamiento de MO debido a que son claras y sin ambigüedades. El hecho de producir una especificación formal obliga a analizar detenidamente los detalles del comportamiento. Por lo tanto, también se puede utilizar como herramienta para identificar y corregir ambigüedades que pudieran pasar desapercibidas en una especificación basada exclusivamente en lenguaje natural. Por estas razones, una especificación formal puede ser útil para mejorar la especificación de comportamiento.

El anexo incluye un ejemplo ilustrativo que muestra la mejor práctica actual. Tiene como objeto establecer una base y un entendimiento comunes de este planteamiento formal en particular que ayudará a lograr coherencia en desarrollos similares. Proporciona un punto de partida útil para usuarios GDMO que deseen utilizar Z para mejorar sus especificaciones de comportamiento.

Está destinado a una audiencia familiarizada con los conceptos básicos de la especificación de objeto gestionado que utilizan plantillas GDMO y con el lenguaje Z.

Para el resto de este anexo, los términos «objeto gestionado» y «MO» se utilizarán para hacer referencia a una definición de clase de objeto gestionado dada que utiliza plantillas GDMO.

B.2 Temas lingüísticos

La notación Z es una notación de especificación formal basada en teoría de conjuntos y en cálculos de predicado. Tiene suficiente capacidad expresiva para poder describir clases únicas de objetos gestionados.

Sin embargo, no existe noción de encapsulamiento en Z. Una especificación Z está constituida normalmente por un modelo de algún estado y por una colección de operaciones para modificar el estado. No existe método construido en Z que parele el estado y sus operaciones en un único módulo y que lo vuelva a utilizar en otra especificación. La consecuencia de esto es aparente cuando se precisa describir objetos gestionados que heredan variables y comportamiento de otras definiciones de clase de objeto gestionado.

El efecto de herencia puede lograrse mediante técnicas de inclusión de esquemas a expensas de algo de claridad. En todos los demás aspectos, Z es adecuado para expresar clases únicas de objetos gestionados.

B.3 Qué es preciso traducir

Las definiciones de comportamiento, o alguna de sus partes, se deben traducir a partir de la descripción informal a Z. Hasta dónde deben formalizarse las partes restantes de las plantillas GDMO depende en gran medida de las necesidades del especificador.

Las plantillas GDMO ya incluyen una definición semiformal de tipos de datos en ASN.1. Es posible escribir una especificación Z utilizando estas definiciones ASN.1 como base para tipos utilizados en la especificación Z, lo que ahorra una cantidad significativa de trabajo.

Sin embargo, escribir una especificación de esta manera supone una tarea mayor para que el especificador esté seguro de que es sintácticamente correcta. Sin especificaciones Z de las definiciones ASN.1, es imposible utilizar los útiles Z existentes que proporcionan soporte para la comprobación de las sintaxis y la semántica estática de una especificación Z.

En resumen, es posible mejorar las definiciones de comportamiento utilizando Z sin volver a escribir los tipos de datos ASN.1, pero se puede obtener un beneficio significativo mediante una traducción completa de los tipos de datos ASN.1 en Z. En B.7.1 se dan ejemplos de cómo convertir tipos básicos ASN.1 en Z.

B.3.1 Desde plantillas GDMO a Z

Esta subcláusula incluye directrices generales de cómo proceder para traducir un objeto gestionado a partir de su descripción informal como la de esta Recomendación | Norma Internacional a Z. Hay que destacar desde el principio que este tipo de traducción sólo puede llevarse a cabo informalmente puesto que una traducción formal requeriría, como mínimo, que tanto el lenguaje fuente como el de destino fueran formales.

Es más, como con cualquier correspondencia entre dos lenguajes distintos, siempre habrá algún desajuste entre sus constructivos. El problema se multiplica cuando ocurre que uno de los lenguajes es informal o incluye componentes informales.

En esta subcláusula se enumeran algunas de las características principales de las plantillas definidas en la presente Recomendación | Norma Internacional junto con las maneras en las que difieren de o se corresponden con constructivos en Z. En el proceso, se indican maneras generales para resolver el desajuste o consejos de cómo pueden tratarse individualmente sobre una base adecuada.

El presente anexo se centrará en lo que es necesario para describir el comportamiento de un objeto gestionado. En B.6 se proporciona información adicional de cómo convertir tipos ASN.1.

B.3.2 Tipos de datos

El primer paso consiste en volver a escribir los tipos de datos a partir de esta Recomendación | Norma Internacional como tipos Z. ASN.1 proporciona las facilidades usuales para escribir datos pero sus constructivos son más adecuados para la descripción de flujos de datos que se comunican entre sistemas.

En ASN.1, los constructivos de tipo se definen como formas de lista. En Z, los tipos son conjuntos. Aunque es posible modelar los constructivos de tipo ASN.1 como secuencias en Z, resulta algunas veces más natural considerar las operaciones disponibles en los tipos ASN.1 y hacer la correspondencia con tipos Z que describen con mayor claridad su estructura. Se pueden hacer corresponder los tipos sequence y set ASN.1 con tuplas Z. El tipo sequence-of ASN.1 se puede hacer corresponder con una secuencia Z. El tipo set-of ASN.1 se puede hacer corresponder con un conjunto Z.

ASN.1 incluye soporte especial para la codificación, por ejemplo etiquetas de tipo y valores por defecto. Esto no es necesario que se represente en Z ya que no afecta a la definición de comportamiento.

La subcláusula B.6.2 da información adicional sobre cómo convertir tipos ASN.1.

B.3.3 Atributos de MO

Los objetos gestionados se definen para que tengan ciertos atributos de gestión. Estos atributos tienen un tipo de datos definido en ASN.1. Son identificadores de objetos asignados. También pueden tener una propiedad de concordancia. Se han propuesto dos maneras de modelizar estos atributos:

- tipos de atributo simple; y
- tipos de atributo como esquemas.

Lo más sencillo consiste en representar el atributo MO en el MO como una variable Z con el tipo de datos apropiado. Por otra parte necesitaremos una definición de constante que represente al identificador de objeto para ese atributo. Esta constante se relacionará con el atributo real sólo mediante convenio. Podemos utilizar la propiedad fija de concordancia cuando se definan para este atributo operaciones de concordancia. En B.6.3 se da un ejemplo.

También es posible encapsular todas estas propiedades de un atributo en un único tipo de esquema que es entonces el tipo de la variable Z que modela el atributo MO. De esta manera, el esquema incluye el valor del atributo, así como el identificador de objeto y la propiedad de concordancia en su caso. En B.6.4 se da un ejemplo. Cuando se requieren reglas de concordancia distintas de la igualdad, es posible definir el parámetro concordancia como una relación Z respecto al tipo del valor del atributo. De esta manera es posible la representación formal de reglas discretionales de concordancia ad hoc, lo que puede ser importante a efectos de fijación de alcance, filtrado y selección de objetos.

Resulta difícil modelar el tipo ASN.1 ANY en Z. Un caso en el que esto es normal consiste en dar listas de valores de atributo. Así, un modelo totalmente formal precisará probablemente un tipo Z libre que combine los tipos de atributo ya definidos. En B.6.1 y B.6.5 puede encontrarse un ejemplo.

Los identificadores de objeto se modelan formalmente mediante un conjunto dado.

[OBJECTID]

B.3.4 Otros identificadores de objeto

Muchas cosas además de los atributos tienen también un identificador de objeto. Resulta conveniente introducirlos como constantes en definiciones axiomáticas. Se utilizará el convenio de añadirles un sufijo «Oid». Normalmente se necesitará este tipo de constantes para clases, lotes y notificaciones.

Un ejemplo es:

```
packagesPackageOid : OBJECTID  
allomorphsPackageOid : OBJECTID  
topClassOid : OBJECTID
```

B.3.5 Herencia y compatibilidad

Z puede utilizarse para construir jerarquías de herencia de MO, utilizando la inclusión de un esquema para modelar la herencia y la especialización. Esto modela correctamente el comportamiento de una clase MO y de sus subtipos pero no hace explícita la fuerte relación de subtipos que está realmente presente. Para ello, se precisa un lenguaje que modele explícitamente la herencia.

De esta manera, Z puede utilizarse para definir satisfactoriamente MO individuales, pero para lograr la consideración de la herencia y de la compatibilidad se precisa la capacidad suplementaria de un lenguaje que modele explícitamente la herencia.

La herencia no está soportada por Z. Puede modelarse mediante la inclusión de esquema simple de esquemas de estado.

La definición de herencia de MO requiere que las subclases sean compatibles. Desgraciadamente, ello no implica que las subclases sean subtipos en Z. Así, normalmente el MO puede indicar su clase real. Puesto que el atributo de clase real siempre informa de una clase real de objeto, una subclase no puede informar de la clase de una superclase. Por lo tanto, una subclase no puede mostrar el mismo comportamiento que su superclase, al devolver el valor de su atributo clase real (es decir, no es sustituible), incluso aunque se esté comportando alomórficamente. Por lo tanto, la división en subclases de clases de objeto gestionado no es equivalente a subtipos Z, en los que un subtipo mostraría el mismo comportamiento que su supertipo. Sin embargo, una subclase muestra muy poco comportamiento «no sustituible».

De esta manera se puede ver que la herencia de MO, según se define en esta Recomendación | Norma Internacional, permite un comportamiento específico en un progenitor que no es coherente con el comportamiento de sus descendientes. Puesto que hay una cantidad muy limitada de este comportamiento no sustituible, una clase MO se puede representar mediante dos especificaciones de clase. Una captura el comportamiento que debe exhibir cualquier instancia y también cualquier MO extendido. La otra es una especialización y captura aquel comportamiento exhibido únicamente por instancias de la clase compatible y no por cualquier extensión. Es esta última especificación la que se considera para dar el comportamiento completo de una instancia de MO real.

B.3.6 Lotes

Muchas partes de la funcionalidad de una clase están presentes en algunos MO individuales y no en otros. Esta Recomendación | Norma Internacional describe este proceso agrupando la funcionalidad en lotes condicionales. Así cada MO ejemplifica lotes adecuados. En Z no se puede proporcionar funcionalidad de esta manera condicional, pero es posible hacer depender el comportamiento del MO de aquellos lotes que están ejemplificados. Esto es inmediato puesto que los MO deben incluir un atributo de gestión denominado lotes que enumera los identificadores de objeto de los lotes realmente ejemplificados. Así, para modelar el comportamiento en un lote condicional, el propio comportamiento se torna condicional en presencia del identificador de lote en el atributo lotes.

B.3.7 Clase

Para definir una clase de MO es necesario representar sus atributos y sus operaciones. Los atributos serán parte del esquema de estados Z y las operaciones serán esquemas de operación Z.

B.3.7.1 Atributos

Los atributos del objeto gestionado se declaran en un esquema de estado. Cada atributo viene dado por un tipo, que puede ser un tipo declarado en la parte ASN.1 de la plantilla GDMO, o que puede utilizar tipos declarados en Z en un modelo totalmente formal.

B.3.7.2 Operación obtención (Get)

El gestor puede solicitar que se realice una operación obtención en un MO. La definición CMISE de M-Get tiene muchos parámetros, pero la mayoría de ellos están relacionados con el control de accesos, la selección de objetos, etc. En esta instancia, Get puede ser modelado en la frontera de objeto gestionado ignorando estos temas y sustituyendo la operación Get única por un número de operaciones Get<name>, en el que <name> es un atributo único.

B.3.7.3 Operación obtención todo (GetAll)

También se modela una operación GetAll, que no tiene entrada. Devuelve un conjunto no vacío de valores de atributo.

B.3.7.4 Operaciones sustitución

La operación Set en un MO individual se solicita mediante una operación M-Set CMISE. Esta especificación modela las operaciones sustitución como se ha visto en la frontera de MO. En esta especificación, las operaciones sustitución hacen referencia a las operaciones de atributo fijación, fijación por defecto, adición y supresión.

La consecuencia de esto es que se especifica un esquema Z para representar cada modificación.

B.3.7.5 Notificaciones

Las notificaciones son mensajes no solicitados enviados por el MO para informar de eventos en su interior. Sin embargo, no se modelan como operaciones. En cambio se modelan como salidas de operaciones que ocurren en el MO. Así, cualquier operación (invocada por el gestor o internamente por el recurso) puede generar una salida y, si provoca una notificación, esta notificación debe ser parte de aquella salida de operación.

Esto significa que la salida de un esquema de operación Z que puede provocar notificaciones debe ser un *conjunto* de notificaciones. Así, aquellas ocasiones en las que no emite una notificación, pueden representarse entregando un conjunto vacío como salida.

Los datos en una notificación consisten en un EventType que es un identificador de objeto de su definición normalizada. A esto sigue información diversa relacionada con esa notificación en particular. El identificador de objeto puede normalmente definirse como una constante y los datos particulares como un tipo esquema. El comportamiento de la notificación está incluido en cualquier objeto que pueda generar la notificación.

B.3.7.6 Acciones

Las acciones son operaciones realizadas por el gestor en un MO. Están representadas muy naturalmente por operaciones Z

B.3.8 Especificación de los sistemas de objetos

En el resto del anexo se describe cómo se representa un objeto único. Cuando se considera la creación/supresión de objetos, las vinculaciones de nombres, la contención y denominación es necesario describir el estado del sistema en que residen los objetos. La creación y supresión de objetos se puede representar mediante un cambio de estado de ese sistema. La vinculación de nombres y la contención se pueden representar mediante una relación respecto al conjunto de objetos. La denominación se puede definir a continuación en términos de esa relación.

B.4 Un ejemplo

En esta subcláusula se dan definiciones de ejemplo para los atributos de clase de MO cima (top) y gestión de estados (State Management). Puesto que el mayor interés de esta guía está en la modelización del comportamiento, no se presenta en esta subcláusula la creación de tipos Z a partir de tipos ASN.1. En B.7 se da una definición formal completa.

B.4.1 top

La primera clase que se define es *top*, que es el último progenitor (en la jerarquía de herencia) de todos los MO.

top tiene cuatro atributos de gestión, *objectClass*, *packages*, *allomorphs* y *nameBinding*. *objectClass* tiene el identificador de objeto de la clase, mientras *packages* tiene los identificadores de objetos de los lotes que ejemplifica. *nameBinding* tiene el identificador de objeto de la ligazón de nombre utilizada para ejemplificar el objeto y *allomorphs* tiene los identificadores de objetos de las clases a las cuales el objeto puede ser alomórfico. Puesto que los atributos de gestión pueden estar en lotes, los atributos presentes en los MO de una clase dada pueden variar. Esto se modela incluyendo un atributo de modelado adicional denominado *attributes*, que tiene los identificadores de objeto de los atributos que están realmente ejemplificados en el MO individual. Hay que destacar que todos los atributos presentes en *top* están fijados durante toda la vida de cualquier MO individual.

Z no modela interfaces explícitamente y, por lo tanto, no es posible definir formalmente qué operaciones están invocadas internamente o externamente por el gestor.

TopState

<p><i>allomorphs</i>: ⊕ OBJECTID <i>objectClass</i>: OBJECTID <i>nameBinding</i>: OBJECTID <i>packages</i>: ⊕ OBJECTID <i>attributes</i>: ⊕ OBJECTID</p>
<p>{<i>objectClassOid</i>, <i>nameBindingOid</i>} ⋈ <i>attributes</i> <i>allomorphsPackageOid</i> ⋈ <i>packages</i> ⊕ <i>allomorphsOid</i> ⋈ <i>attributes</i> <i>packagesPackageOid</i> ⋈ <i>packages</i> <i>packages</i> ⊕ ⋈ ⊕ <i>packagesOid</i> ⋈ <i>attributes</i></p>

attributes no es un atributo de MO, sino un nuevo componente de estado definido por conveniencia. Enumera los atributos de MO que contiene un MO. Así el invariante obliga a incluir los identificadores de objetos de los atributos pertinentes como se ha descrito en B.3.3 (y se ha definido en B.7.4). *objectClass* y *nameBinding* son obligatorios. *packages* está presente si se ejemplifica cualquier lote registrado que no sea *packagesPackage*. En este caso esto significa *allomorphsPackage*.

La operación *TopGetNameBinding* interroga al MO y devuelve el valor del atributo *nameBinding*, sin cambiar *TopState*. *TopGetNameBinding* es invocado por el gestor.

TopGetNameBinding

<p>⋈ <i>TopState</i> <i>result!</i>: OBJECTID</p>
<p><i>result!</i> = <i>nameBinding</i></p>

No se han definido aquí las operaciones *TopGetAllomorphs*, *TopGetObjectClass* y *TopGetPackages*. Hay que destacar que no existe operación para obtener *attributes*, puesto que *attributes* no es un atributo de MO real como se especifica en la plantilla GDMO.

TopGetAll obtiene todos los valores de atributo de un objeto. Siempre devuelve valores para *objectClass* y *nameBinding*. Si están presentes lotes condicionales o alomorfos, también obtiene éstos. *TopGetAll* es invocado por el gestor.

TopGetAll

✕ *TopState*
result!:
 ⊕ *AttributeValues*

attributes = # result!
ObjectClassValue objectClass ∂ *result!*
NameBindingValue nameBinding ∂ *result!*
PackagesOid ∂ *attributes* ∪ *packagesValue packages* ∂ *result!*
AllomorphsOid ∂ *attributes* ∪ *allomorphsValue allomorphs* ∂ *result!*

TopEventReport es una manera de modelar notificaciones. *TopEventReport* aparece espontáneamente y representa la manera en que los informes de evento no están controlados por el gestor.

TopEventReport

✕ *TopState*
notification!: *EventInfo*

B.4.2 Clase StateManagement

Esta clase no refleja ninguna clase específica de MO. En cambio, refleja el comportamiento de cualquier objeto que contenga cualesquiera atributos de una cierta norma: *administrativeState*, *operationalState* y *usageState*. Resulta más conveniente en este marco entender esta inclusión como herencia, lo que sirve como un ejemplo útil.

El esquema de estado incluye las definiciones y predicados *TopState*, y define algunas variables y conjunciones de predicado adicionales.

StateManagementState

TopState
administrativeState:
AdministrativeState
operationalState: *OperationalState*
usageState: *UsageState*

operationalState = disabled ∪ *usageState = idle*
administrativeState = locked ∪ *usageState = idle*
usageState = idle ∪ *administrativeState* ⊕ *shuttingDown*

State Management hereda las operaciones de Top. Aunque no existe mecanismo incluido en Z para operaciones de herencia, resulta inmediato volver a definir las operaciones en los términos del nuevo estado. La parte predicado de *StateManagementState* se obtiene de la definición de la función de gestión de estados de la Rec. X.721 del CCITT (1992) | ISO/CEI 10165-2:1992 y de la Rec. X.731 del CCITT (1992) | ISO/CEI 10164-2:1992.

La operación *SMGetNameBinding* puede definirse fácilmente, puesto que no tiene efecto en las nuevas variables de estado declaradas en *StateManagementState*. Se puede volver a utilizar la definición de *TopGetNameBinding*:

SMGetNameBinding

TopGetNameBinding
 ✕ *StateManagementState*

También se han omitido de este ejemplo las definiciones para operaciones para obtener los demás atributos de *StateManagementState*. Las operaciones *SMGetAllomorphs*, *SMGetObjectClass* y *SMGetPackages* pueden reutilizar las definiciones a partir de *Top* como en *SMGetNameBinding*. Será preciso definir nuevas operaciones para *GetSMAAdministrativeState*, *GetSMOperationalState* y *SMGetUsageState*. *SMEventReport* también puede reutilizarse.

El esquema *SMGetAll* también utiliza una operación definida en *TopState*. Incluye la definición de *TopGetAll* y refuerza la condición posterior.

SMGetAll

<p>⊗ <i>StateManagementState</i> <i>TopGetAll</i></p>
<p><i>administrativeStateOid</i> ⊗ <i>attributes</i> ⊕ <i>administrativeStateValue</i> <i>administrativeState</i> ⊗ <i>result!</i> <i>OperationalStateOid</i> ⊗ <i>attributes</i> ⊕ <i>operationalStateValue</i> <i>operationalState</i> ⊗ <i>result!</i> <i>UsageStateOid</i> ⊗ <i>attributes</i> ⊕ <i>usageStateValue</i> <i>usageState</i> ⊗ <i>result!</i></p>

La operación *SMReplaceAdministrativeState* describe un comportamiento específico para la clase de gestión de estados, mientras que el estado administrativo se sustituye por otro valor suministrado como una entrada. En función del estado del objeto cuando se lleva a cabo la operación, el estado de utilización puede también modificarse. El estado operacional no se altera por esta operación.

SMReplaceAdministrativeState

<p>⊗ <i>StateManagementState</i> ⊗ <i>TopState</i> <i>input?: AdministrativeState</i></p>
<p><i>administrativeState'</i> ⊗ IF <i>usageState</i> ⊕ <i>idle</i> THEN { <i>unlocked</i> ⊗ <i>unlocked</i>, <i>locked</i> ⊗ <i>locked</i>, <i>shuttingDown</i> ⊗ <i>locked</i>, <i>locked</i> ⊗ <i>shuttingDown</i>, <i>shuttingDown</i> ⊗ <i>shuttingDown</i> } ⤴ { <i>input?</i> } ⤴ ELSE { <i>unlocked</i> ⊗ <i>unlocked</i>, <i>locked</i> ⊗ <i>locked</i>, <i>shuttingDown</i> ⊗ <i>locked</i> } ⤴ { <i>input?</i> } ⤴ <i>administrativeState'</i> = <i>locked</i> ⊕ <i>usageState'</i> = <i>idle</i> <i>administrativeState'</i> ⊕ <i>locked</i> ⊕ <i>usageState'</i> = <i>usageState</i> <i>operationalState'</i> = <i>operationalState</i></p>

El comportamiento especificado en la parte de predicado del esquema es una formalización de la descripción informal según la Rec. X.731 del CCITT | ISO/CEI 10164-2. Para ser completos, se deberían también definir las operaciones para sustituir el estado operacional y el estado de utilización.

Finalmente existen algunas otras operaciones que describen el comportamiento específico de la clase de gestión de estados. Estas operaciones no se enumeran aquí, aunque pueden encontrarse en B.7. Estas operaciones incluyen *SMCapacityDecrease*, *SMCapacityIncrease*, *SMDisable*, *SMAEnable*, *SMNewUser* y *SMUserQuit*.

B.4.3 Clases ejemplificables

Ninguna de las clases descritas anteriormente pueden ejemplificarse. El procedimiento que se ha seguido puede continuar. *StateManagement* se puede volver a utilizar para definir una clase denominada *CIRCUIT*, que a su vez puede utilizarse para definir *ECIRCUIT* y de esta manera la clase ejemplificable *ActualECircuit*.

Esto se ha omitido de la guía, puesto que los procedimientos son exactamente los mismos que se han destacado y su repetición no añade nada.

B.5 Temas pendientes

En esta subcláusula, se enumeran los temas principales encontrados durante la traducción en Z de las especificaciones de objetos gestionados basados en GDMO. Cuando un tema se relaciona con Z sin tener el constructivo correspondiente para una característica particular de la especificación de objetos gestionados, el tratamiento informal utilizado que se propone se incluye también en este anexo.

B.5.1 Definición de comportamiento en objetos gestionados

En las plantillas, el término «definición de comportamiento» (*'behaviour definition'*) se utiliza para prácticamente todas las entidades ya sean datos o procesos. En este último caso, pueden incluir información sobre el comportamiento real (en un sentido estricto), o solamente información estática sobre la entidad, como la utilización que se pretende, o ambas cosas. Cuando se traduce, es preciso analizar el texto que se encuentra bajo este encabezamiento y extraer la información relativa al comportamiento para la entidad de que se trate. Esta información de comportamiento se utilizará en la traducción formal, mientras que el texto real se puede incluir como un comentario dentro de la especificación Z.

B.5.2 Operaciones internas en Z

Una operación interna en un objeto gestionado representa el caso en el que una notificación se emite espontáneamente (sin que implique ninguna invocación de gestión). Las operaciones internas son también un hecho deseado en muchos otros sistemas. Actualmente, en Z, este hecho se representa informalmente mediante un comentario en el texto de lenguaje natural que es un hecho importante de cualquier especificación Z correctamente escrita.

B.5.3 Clases abstractas en Z

Algunas veces resulta útil identificar clases abstractas: es decir, clases sin ejemplificaciones propias. Algunas clases de MO (como *top*) no pueden ejemplificarse. Sería muy útil poder mostrar qué partes de las especificaciones Z correspondientes representan clases que pueden ejemplificarse. Hasta ahora esto se ha tenido en cuenta mediante una anotación informal.

B.5.4 Semántica PARAMETER

En esta Recomendación | Norma Internacional no se considera la incorporación de la semántica PARAMETER en los objetos.

B.6 Conversión de tipos de datos ASN.1 en Z

Los elementos para la traducción se describirán para cada constructivo ASN.1.

B.6.1 Tipos simples

ASN.1 incluye algunos tipos simples que están contruidos dentro. Éstos tienen una estructura normalizada pero normalmente no es interesante en el contexto de estas especificaciones, por lo que pueden representarse fundamentalmente como conjuntos dados. Existe una amplia variedad de tipos de cadenas de caracteres.

[*NUMERICSTRING, PRINTABLESTRING, TELETEXSTRING, VIDEOTEXSTRING, VISIBLESTRING, IA5STRING, GRAPHICSTRING, GENERALSTRING*]

Dos de ellos tienen sinónimos.

T61STRING == *TELETEXSTRING*

ISO64STRING == *VISIBLESTRING*

De los otros tipos simples, Integer puede representarse por ☺, Boolean y Null mediante tipos libres:

Boolean ::= *btrue* | *bfalse*

Null ::= *null*

Obsérvese que estos tipos libres también definen la notación de valor para estos tipos.

Real, Bit String y Octet String pueden habitualmente considerarse como conjuntos dados (aunque algunas veces puede ser necesario estructurar los tipos Bit y Octet String).

[*REAL,BITSTRING,OCTETSTRING*]

Esta Recomendación | Norma Internacional también describe otro tipo especial que se proporcionará como un conjunto dado.

[*OBJECTID*]

Aquí *OBJECTID* representa un identificador de objeto ASN.1.

Los identificadores de objeto son de hecho secuencias no vacías de \mathbb{Z} y puede ser conveniente modelarlas como tales, en lugar de como un conjunto dado. En este caso debe pensarse con cuidado la notación de valor adecuada.

Existen también algunos tipos «útiles» que están definidos en ASN.1 dentro de la norma ASN.1. Así, aunque podrían estar definidos en términos de los demás constructivos ASN.1, es conveniente de nuevo suministrarlos como conjuntos dados.

[*GENERALIZEDTIME, UTCTIME, OBJECTDESCRIPTOR, EXTERNAL*]

Any

ASN.1 admite un tipo especial ANY que puede contener cualquier otro tipo ASN.1. Esta clase de tipo no está permitida en Z y sería difícil ampliarlo para incluirlo. Sin embargo, dado cualquier conjunto conocido de tipos, es posible definir un tipo libre Z que pueda incluir cualquiera de estos otros tipos. Una estrategia alternativa consiste en definir ANY como un conjunto dado con la finalidad de comprobación de tipo. Esto resulta satisfactorio siempre que no se haga nada más con él. El tipo *AttributeValues* normalmente sustituye a ANY. Esto se define más adelante.

B.6.2 Tipos estructurados

Otros tipos en ASN.1 están contruidos por constructivos.

Set

Conjuntos ASN.1 pueden representarse como tuplas o esquemas en Z. Las tuplas Z no permiten denominar componentes y por lo tanto los esquemas pueden ser más apropiados. Sin embargo, la notación de valor Z para esquemas resulta menos conveniente. El etiquetado no sólo no es necesario sino que no es posible en Z, puesto que los componentes de un «conjunto» pueden siempre ser discriminados por su posición en una tupla o por su denominación de componente en un esquema.

Los componentes en este o en otros tipos estructurados pueden ser *OPTIONAL*. Esto se puede representar en Z aumentando el tipo del componente opcional con un valor especial «ausente». Los valores *DEFAULT* no se pueden representar adecuadamente como un elemento de un tipo de datos. Es posible representar el comportamiento que implica default dentro de cualquier operación sobre estos datos.

Sequence

Se pueden modelar secuencias ASN.1 exactamente de la misma manera que los conjuntos ASN.1, puesto que la única diferencia es que no existe orden explícito. Dado que éste es el caso, podría argumentarse que una tupla resulta más adecuada, pero los esquemas también pueden utilizarse.

Set-of

Los tipos Set-of ASN.1 son realmente bolsas y se pueden definir en Z como tales. Se debe destacar que MIM requiere explícitamente que todo este tipo de bolsas se traten como conjuntos y por lo tanto resulta de hecho más apropiado modelar el tipo como un conjunto Z.

Cuando se precisa hacer subtipo a un tipo ASN.1, es normalmente necesario añadir una restricción de predicado al tipo. En algunos casos, por ejemplo para los subtipos entero y esquema, se puede hacer en la propia definición de tipo. En otros casos (por ejemplo subtipos bolsa) la restricción debe aplicarse a variables definidas pertenecientes a dicho tipo.

Sequence-of

Los tipos Sequence-of ASN.1 se pueden modelar adecuadamente como secuencias Z.

Choice

Los tipos Choice ASN.1 son enumeraciones directas y se pueden modelar mediante tipos libres Z.

Este tipo introduce un problema serio de ámbito. En ASN.1 los constructivos dentro de Choice son locales a este tipo. Por lo tanto, se puede utilizar un único nombre de constructivo en más de una enumeración. En Z estos nombres son globales y no se pueden volver a utilizar. Este problema se debe normalmente resolver cambiando los nombres de los constructivos, mediante un prefijo con el nombre de su tipo.

Un problema similar surge cuando se generan tipos ASN.1 que son sinónimos de Integer (por ejemplo) pero con valores denominados. Estos valores denominados son locales al tipo sinónimo en ASN.1 pero sinónimos globales para enteros en Z. De nuevo esto se debe arreglar cambiando los nombres de los constructivos.

B.6.3 Tipos de atributo simple

Lo más simple es representar el atributo de MO dentro del MO como una variable Z con el tipo de datos adecuado. Se necesitará una definición de constante que represente el *OBJECTID* de dicho atributo. Cuando se definen operaciones de concordancia para este atributo, se puede utilizar el valor normalizado real del parámetro matches-for.

De esta manera, considérese el atributo de MO *administrativeState*. Habremos definido un tipo:

AdministrativeState ::= unlocked | locked | shuttingDown

Se puede definir una constante para representar el identificador de objeto de atributo:

<i>administrativeStateOid : OBJECTID</i>
--

El valor real del identificador puede presentar una restricción en esa definición axiomática.

Entonces se definirá una variable de estado en el MO:

MOState

<i>administrativeState: AdministrativeState</i>

Esta solución es inmediata y conveniente pero no precisa enumeraciones de definiciones axiomáticas *OBJECTID*. También establece el enlace entre el nombre del atributo y su *OBJECTID* de manera puramente sintáctica. Se ha adoptado el convenio de añadir un sufijo *Oid*.

B.6.4 Tipos de atributo como esquemas

También es posible encapsular estas características de un atributo en un único tipo de esquema que será el tipo de la variable Z que modele el atributo de MO:

AdministrativeStateType

<i>value:AdministrativeState</i>

<i>Oid:OBJECTID</i>

<i>Oid = ↗4, 3, 19, 27, 1, 3 ↖</i>

Es importante proporcionar un valor para el *OBJECTID* puesto que es necesario establecer que no se puede cambiar incluso aunque el valor sí se pueda.

No se ha definido una estructura para *OBJECTID* pero

OBJECTID ::= *seq*₁ ①

es una posibilidad que podría dar sentido al esquema precedente. Este esquema podría también tener el parámetro *matches-for* si se considerara importante representarlo dentro de la especificación.

Así el MO contendría un atributo con este tipo:

MState

administrativeState : *AdministrativeStateType*

La referencia a su valor o a su Oid se realizará mediante la selección adecuada como en *administrativeState.value*.

Esta técnica proporciona de manera adecuada la semántica para la conexión entre un atributo y su *OBJECTID*. Sin embargo, puede parecer extraño a los lectores de la especificación que esté presente Oid en el estado de MO, incluso aunque no pueda cambiar (y de hecho es una constante global conocida en el momento de la especificación).

B.6.5 Tipo AttributeValues

Como se ha mencionado anteriormente, es difícil modelar el tipo ANY de ASN.1 en Z. Un caso en el que esto es habitual consiste en dar listas de valores de atributo. Se necesitará una definición de tipo libre Z que combine los tipos de atributos ya definidos. Este planteamiento funciona siempre que el conjunto de atributos en uso se fije en el momento de la especificación. Entonces normalmente se parecerá a:

AttributeValues ::= *administrativeStateValue* @*AdministrativeState* *O* |
objectClassValue @*OBJECTID* *O* |
nameBindingValue @*OBJECTID* *O* |
packagesValue @ ⊕ *OBJECTID* *O* |
allomorphsValue @ ⊕ *OBJECTID* *O* |
operationalStateValue @*OperationalState* *O* |
usageStateValue @*UsageState* *O*

B.7 Ejemplo completo

Esta subcláusula presenta el modelo formal completo en el que se basa el ejemplo de B.4. Se presenta en el estilo de declaración Z tradicional, antes de su uso: es decir, las definiciones de tipo convertidas desde ASN.1 aparecen al principio y las definiciones del comportamiento aparecen al final.

Hay un punto de la especificación que vale la pena comentar. Las definiciones *AttributeValues* y *OBJECTINSTANCE* son recursivas mutuamente. Esto es técnicamente ilegal en Z y, por lo tanto, se ha hecho lo siguiente para permitir la definición. Se ha introducido *AttributeValues* como un conjunto dado. *OBJECTINSTANCE* se define entonces utilizando el conjunto dado *AttributeValues*. Se utiliza entonces esta definición de *OBJECTINSTANCE* para introducir las restricciones que puede tomar *AttributeValues*. Se ha cargado la obligación de prueba que muestra que estos conjuntos existen realmente, pero no se presenta.

B.7.1 Tipos básicos ASN.1

[*NUMERICSTRING*, *PRINTABLESTRING*, *TELETEXSTRING*, *VIDEOTEXSTRING*]
[*VISIBLESTRING*, *IA5STRING*, *GRAPHICSTRING*, *GENERALSTRING*]

T61STRING ::= *TELETEXSTRING*

ISO64STRING ::= *VISIBLESTRING*

Boolean ::= *btrue* | *bfalse*

Null ::= null

[REAL,BITSTRING,OCTETSTRING]

[OBJECTID]

[ANY]

[GENERALIZEDTIME,UTCTIME,OBJECTDESCRIPTOR,EXTERNAL]

B.7.2 Atributos de MO

El siguiente conjunto dado es una especie de reserva de lugar (placeholder) para una definición de tipo libre más compleja y más completa dada de manera incremental según se van definiendo las nuevas clases.

[AttributeValues]

AttributeValuesOptional ::= present @ AttributeValues O / absent

RelativeDistinguishedName == AttributeValues

RDNSequence == seq RelativeDistinguishedName

DistinguishedName == RDNSequence

*OBJECTINSTANCE ::= distinguishedName @DistinguishedName O / nonSpecificForm @[⊕]O
/ localDistinguishedName @RDNSequence O*

B.7.3 Notificaciones

ProbableCause == OBJECTID

SpecificIdentifier ::= globalvalue @OBJECTID O / localValue @[⊕]O

SpecificProblems == [⊕] SpecificIdentifier

SpecificProblemsOptional ::= sPPresent @ SpecificProblems O / sPAbsent

PerceivedSeverity ::= indeterminate | critical | major | minor | warning | cleared

BackedUpStatus == Boolean

BackedUpStatusOptional ::= bUSPresent @ BackedUpStatus O / bUSAbsent

ObjectInstanceOptional ::= oIPresent @ OBJECTINSTANCE O / oIAbsent

ISO/CEI 10165-4 : 1992/enm.3 : 1998 (S)

TrendIndication ::= *lessSevere* | *noChange* | *moreSevere*

TrendIndicationOptional ::= *tIPresent* @ *TrendIndication* ○ | *tIAbsent*

ObservedValue ::= *int* @ $\textcircled{\ominus}$ ○ | *real* @ *REAL* ○

ObservedValueOptional ::= *oVPresent* @ *ObservedValue* ○ | *oVAbsent*

ThresholdLevelInd ::=
 up @ *ObservedValue* \diamond *ObservedValueOptional* ○
 | *down* @ *ObservedValue* \diamond *ObservedValueOptional* ○

ThresholdLevelIndOptional ::= *tLIPresent* @ *ThresholdLevelInd* ○ | *tLIAbsent*

ArmTimeOptional ::= *aTPresent* @ *GENERALIZEDTIME* ○ | *aTAbsent*

ThresholdInfo ==
 OBJECTID \diamond *ObservedValue* \diamond *ThresholdLevelIndOptional* \diamond *ArmTimeOptional*

ThresholdInfoOptional ::= *thIPresent* @ *ThresholdInfo* ○ | *thIAbsent*

NotificationIdentifier == $\textcircled{\ominus}$

NotificationIdentifierOptional ::= *nIPresent* @ *NotificationIdentifier* ○ | *nIAbsent*

CorrelatedNotifications == $\textcircled{\oplus}$ (($\textcircled{\oplus}$ *NotificationIdentifier*) \diamond *ObjectInstanceOptional*)

CorrelatedNotificationsOptional ::= *cNPresent* @ *CorrelatedNotifications* ○ | *cNAbsent*

AttributeValueChangeDefinition == $\textcircled{\oplus}$ (*OBJECTID* \diamond *AttributeValuesOptional* \diamond *AttributeValues*)

AttributeValueChangeDefinitionOptional ::=
 aVCDPresent @ *AttributeValueChangeDefinition* ○ | *aVCDAbsent*

MonitoredAttributes == $\textcircled{\oplus}$ *OBJECTID*

MonitoredAttributesOptional ::= *mAPresent* @ *MonitoredAttributes* ○ | *mAbsent*

ProposedRepairActions == $\textcircled{\oplus}$ *SpecificIdentifier*

ProposedRepairActionsOptional ::= *pRAPresent* @ *ProposedRepairActions* ○ | *pRAbsent*

AdditionalTextOptional ::= *adTPresent* @ *GRAPHICSTRING* ○ | *adTAbsent*

ManagementExtension == *OBJECTID* \diamond *Boolean* \diamond *ANY*

AdditionalInformation == $\textcircled{\oplus}$ *ManagementExtension*

AdditionalInformationOptional ::= *aIPresent* @ *AdditionalInformation* *O* / *aIAbsent*

SourceIndicator ::= *resourceOperation* / *managementOperation* / *sIUnknown*

SourceIndicatorOptional ::= *sIPresent* @ *SourceIndicator* *O* / *sIAbsent*

AttributeIdentifierList == ⊕ *OBJECTID*

AttributeIdentifierListOptional ::= *atIPresent* @ *AttributeIdentifierList* *O* / *atIAbsent*

Attribute == *OBJECTID* ⋄ *AttributeValues*

AttributeList == ⊕ *Attribute*

AttributeListOptional ::= *aLPresent* @ *AttributeList* *O* / *aLAbsent*

AlarmInfo

probableCause: *ProbableCause*
specificProblems: *SpecificProblemsOptional*
perceivedSeverity: *PerceivedSeverity*
backedUpStatus: *BackedUpStatusOptional*
backUpObject: *ObjectInstanceOptional*
trendIndication: *TrendIndicationOptional*
thresholdInfo: *ThresholdInfoOptional*
notificationIdentifier: *NotificationIdentifierOptional*
correlatedNotifications: *CorrelatedNotificationsOptional*
stateChangeDefinition: *AttributeValueChangeDefinitionOptional*
monitoredAttributes: *MonitoredAttributesOptional*
proposedRepairActions: *ProposedRepairActionsOptional*
additionalText: *AdditionalTextOptional*
additionalInformation: *AdditionalInformationOptional*

AttributeValueChangeInfo

sourceIndicator: *SourceIndicatorOptional*
attributeIdentifierList: *AttributeIdentifierListOptional*
attributeValueChangeDefinition: *AttributeValueChangeDefinitionOptional*
notificationIdentifier: *NotificationIdentifierOptional*
correlatedNotifications: *CorrelatedNotificationsOptional*
additionalText: *AdditionalTextOptional*
additionalInformation: *AdditionalInformationOptional*

ObjectInfo

sourceIndicator: *SourceIndicatorOptional*
attributeList: *AttributeListOptional*
notificationIdentifier: *NotificationIdentifierOptional*
correlatedNotifications: *CorrelatedNotificationsOptional*
additionalText: *AdditionalTextOptional*
additionalInformation: *AdditionalInformationOptional*

RelationshipChangeInfo

sourceIndicator: SourceIndicatorOptional
attributeIdentifierList: AttributeIdentifierListOptional
relationshipChangeDefinition: AttributeValueChangeDefinitionOptional
notificationIdentifier: NotificationIdentifierOptional
correlatedNotifications: CorrelatedNotificationsOptional
additionalText: AdditionalTextOptional
additionalInformation: AdditionalInformationOptional

SecurityAlarmInfo

notificationIdentifier: NotificationIdentifierOptional
correlatedNotifications: CorrelatedNotificationsOptional
additionalText: AdditionalTextOptional
additionalInformation: AdditionalInformationOptional

StateChangeInfo

sourceIndicator: SourceIndicatorOptional
attributeIdentifierList: AttributeIdentifierListOptional
stateChangeDefinition: AttributeValueChangeDefinitionOptional
notificationIdentifier: NotificationIdentifierOptional
correlatedNotifications: CorrelatedNotificationsOptional
additionalText: AdditionalTextOptional
additionalInformation: AdditionalInformationOptional

EventInfo ::= attributeValueChange @AttributeValueChangeInfo ○
/ communicationsAlarm @AlarmInfo ○
/ environmentalAlarm @AlarmInfo ○
/ equipmentAlarm @AlarmInfo ○
/ integrityViolation @SecurityAlarmInfo ○
/ objectCreation @ObjectInfo ○
/ objectDeletion @ObjectInfo ○
/ operationalViolation @SecurityAlarmInfo ○
/ physicalViolation @SecurityAlarmInfo ○
/ processingError @AlarmInfo ○
/ qualityOfServiceAlarm @AlarmInfo ○
/ relationshipChange @RelationshipChangeInfo ○
/ securityServiceOrMechanismViolation @SecurityAlarmInfo ○
/ stateChange @StateChangeInfo ○
/ timeDomainViolation @SecurityAlarmInfo ○

B.7.4 «Rec. X.721 del CCITT (1992)» | ISO/CEI 10165-2:1992:Top

allomorphsOid: OBJECTID
nameBindingOid: OBJECTID
objectClassOid: OBJECTID
packagesOid: OBJECTID

allomorphsValue : (⊕ OBJECTID) ⊙ AttributeValues
nameBindingValue : OBJECTID ⊙ AttributeValues
objectClassValue : OBJECTID ⊙ AttributeValues
packagesValue : (⊕ OBJECTID) ⊙ AttributeValues

disjoint ↑**ran** *allomorphsValue*, **ran** *nameBindingValue*,
ran *objectClassValue*, **ran** *packagesValue* ↑

packagesPackageOid: OBJECTID
allomorphsPackageOid: OBJECTID

TopState

allomorphs: ⊕ OBJECTID
objectClass: OBJECTID
nameBinding: OBJECTID
packages: ⊕ OBJECTID
attributes: ⊕ OBJECTID

{ *objectClassOid*, *nameBindingOid* } ℑ *attributes*
allomorphsPackageOid ℑ *packages* ∪ *allomorphsOid* ℑ *attributes*
packagesPackageOid ℑ *packages*
packages ⊕ ↗ ∪ *packagesOid* ℑ *attributes*

TopGetNameBinding

≠*TopState*
result!: OBJECTID

result! = *nameBinding*

TopGetAll

≠*TopState*
result!: ⊕ AttributeValues

attributes = # *result!*
ObjectClassValue *objectClass* ℑ *result!*
NameBindingValue *nameBinding* ℑ *result!*
PackagesOid ℑ *attributes* ∪ *packagesValue* *packages* ℑ *result!*
AllomorphsOid ℑ *attributes* ∪ *allomorphsValue* *allomorphs* ℑ *result!*

TopEventReport

≠*TopState*
notification!: EventInfo

B.7.5 Class StateManagement

administrativeStateOid: OBJECTID
operationalStateOid: OBJECTID
usageStateOid: OBJECTID

AdministrativeState ::= unlocked | locked | shuttingDown

OperationalState ::= enabled | disabled

UsageState ::= idle | active | busy

administrativeStateValue: AdministrativeState @AttributeValues
operationalStateValue: OperationalState @AttributeValues
usageStateValue: UsageState @AttributeValues

disjoint \uparrow *ran allomorphsValue, ran nameBindingValue, ran objectClassValue,*
ran packagesValue, ran administrativeStateValue,
ran operationalStateValue, ran usageStateValue \hat{r}

StateManagementState

TopState
administrativeState: AdministrativeState
operationalState: OperationalState
usageState: UsageState

operationalState = disabled \cup *usageState = idle*
administrativeState = locked \cup *usageState = idle*
usageState = idle \cup *administrativeState* \oplus *shuttingDown*

SMGetNameBinding

TopGetNameBinding
 \neq *StateManagementState*

SMEventReport

TopEventReport
 \neq *StateManagementState*

SMGetAll

\neq *StateManagementState*
TopGetAll

administrativeStateOid \neq *attributes*
 \cup *administrativeStateValue administrativeState* \neq *result!*
OperationalStateOid \neq *attributes* \cup *operationalStateValue operationalState* \neq *result!*
UsageStateOid \neq *attributes* \cup *usageStateValue usageState* \neq *result!*

SMCapacityDecrease

⊗*StateManagementState*
 ≠*TopState*

usageState = active ⊕ *usageState*' ∈ {active, busy}
usageState ⊕ active ⊕ *usageState*' = *usageState*
administrativeState' = *administrativeState*
operationalState' = *operationalState*

SMCapacityIncrease

⊗*StateManagementState*
 ≠*TopState*

usageState = busy ⊕ *usageState*' = active
usageState ⊕ busy ⊕ *usageState*' = *usageState*
administrativeState' = *administrativeState*
operationalState' = *operationalState*

SMDisable

⊗*StateManagementState*
 ≠*TopState*

administrativeState' =
 IF *administrativeState* = *shuttingDown*
 THEN *locked*
 ELSE *administrativeState*
operationalState' = *disabled*
usageState' = *idle*

SMEnable

⊗*StateManagementState*
 ≠*TopState*

operationalState = *disabled*
operationalState' = *enabled*
administrativeState' = *administrativeState*
usageState' = *usageState*

SMNewUser

⊗*StateManagementState*
 ≠*TopState*

operationalState = *enabled*
administrativeState = *unlocked*
usageState ∈ {idle, active}
usageState' ∈ {active, busy}
administrativeState' = *administrativeState*
operationalState' = *operationalState*

SMUserQuit

<p>⊗StateManagementState ⊗TopState</p>	<p>$administrativeState = shuttingDown \wedge usageState' = idle$ $\bigcirc administrativeState' = locked$ $administrativeState \oplus shuttingDown \vee usageState' \oplus idle$ $\bigcirc administrativeState' = administrativeState$ $usageState \mathcal{D}\{ active, busy\}$ $usageState' \mathcal{D}\{ idle, active\}$ $operationalState' = operationalState$</p>
--	--

SMReplaceAdministrativeState

<p>⊗StateManagementState ⊗TopState</p>	<p>$administrativeState' \mathcal{D}$ IF $usageState \oplus idle$ THEN { $unlocked \otimes unlocked, locked \otimes locked,$ $shuttingDown \otimes locked, locked \otimes shuttingDown,$ $shuttingDown \otimes shuttingDown$ } $\wedge \{ input?\} \star$ ELSE { $unlocked \otimes unlocked, locked \otimes locked,$ $shuttingDown \otimes locked$ } $\wedge \{ input?\} \star$ $administrativeState' = locked \bigcirc usageState' = idle$ $administrativeState' \oplus locked \bigcirc usageState' = usageState$ $operationalState' = operationalState$</p>
--	---

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	Transmisiones de señales radiofónicas, de televisión 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
Serie Z	Lenguajes de programación