UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

H.248.1 v2

Corrigendum 1

(03/2004)

SERIE H: SISTEMAS AUDIOVISUALES Y MULTIMEDIOS

Infraestructura de los servicios audiovisuales – Procedimientos de comunicación

---

Protocolo de control de las pasarelas: Versión 2

Corrigendum 1

Recomendación UIT-T H.248.1v2 (2002) – Corrigendum 1

RECOMENDACIONES UIT-T DE LA SERIE H

SISTEMAS AUDIOVISUALES Y MULTIMEDIOS

CARACTERÍSTICAS DE LOS SISTEMAS VIDEOTELEFÓNICOS	H.100–H.199
INFRAESTRUCTURA DE LOS SERVICIOS AUDIOVISUALES	
Generalidades	H.200–H.219
Multiplexación y sincronización en transmisión	H.220–H.229
Aspectos de los sistemas	H.230–H.239
Procedimientos de comunicación	**H.240–H.259**
Codificación de imágenes vídeo en movimiento	H.260–H.279
Aspectos relacionados con los sistemas	H.280–H.299
Sistemas y equipos terminales para los servicios audiovisuales	H.300–H.349
Arquitectura de servicios de directorio para servicios audiovisuales y multimedios	H.350–H.359
Arquitectura de la calidad de servicio para servicios audiovisuales y multimedios	H.360–H.369
Servicios suplementarios para multimedios	H.450–H.499
PROCEDIMIENTOS DE MOVILIDAD Y DE COLABORACIÓN	
Visión de conjunto de la movilidad y de la colaboración, definiciones, protocolos y procedimientos	H.500–H.509
Movilidad para los sistemas y servicios multimedios de la serie H	H.510–H.519
Aplicaciones y servicios de colaboración en móviles multimedios	H.520–H.529
Seguridad para los sistemas y servicios móviles multimedios	H.530–H.539
Seguridad para las aplicaciones y los servicios de colaboración en móviles multimedios	H.540–H.549
Procedimientos de interfuncionamiento de la movilidad	H.550–H.559
Procedimientos de interfuncionamiento de colaboración en móviles multimedios	H.560–H.569
SERVICIOS DE BANDA ANCHA Y DE TRÍADA MULTIMEDIOS	
Servicios multimedios de banda ancha sobre VDSL	H.610–H.619

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

Recomendación UIT-T H.248.1v2

Protocolo de control de las pasarelas: Versión 2

Corrigendum 1

Resumen

Para obtener una mayor escalabilidad, la presente Recomendación descompone la función pasarela H.323, definida en la Rec. UIT-T H.246, en subcomponentes funcionales y especifica los protocolos utilizados por estos componentes para comunicar. Esto permite que las implementaciones de pasarelas H.323 sean escalables en alto grado e incita a servirse de las capacidades ofrecidas por las redes con conmutación de circuitos (RCC), tales como los conmutadores SS7. Esto también permite que las pasarelas H.323 estén formadas por componentes suministrados por múltiples vendedores, distribuidos a través de múltiples plataformas físicas. La presente Recomendación tiene por objeto añadir capacidades actualmente definidas para sistemas H.323, con el fin de proporcionar nuevas formas de efectuar operaciones ya soportadas en la Rec. UIT-T H.323.

En esta Recomendación se han hecho algunas mejoras con respecto a la Versión 1 de la misma Recomendación:

– auditoría específica de estadísticas, propiedades, señales y eventos;

– mejor tratamiento de multiplexación;

– topología de trenes;

– descripción más completa de los perfiles;

– modificación de ServiceChange.

El presente corrigendum 1 subsana varios defectos encontrados en la Recomendación, en concreto:

• Especificación de los tipos para rtp/jit y rtp/delay en E.12.4.

• Definición del símbolo '#' en la construcción "unequal" de la codificación de texto.

• Definición del símbolo en el contexto NULL de la codificación de texto.

• Correcciones a: las estadísticas de ejemplo del apéndice I; las directrices para las estadísticas definidas por el lote de 12.1.5; la auditoría ambigua y el retorno del valor de auditoría individual; retorno del valor Context Audit; un error tipográfico en 7.1.2.

• Especificación del significado del término "automático" en el "lote de circuitos TDM" de E.13.

• Adición de un punto de código y de un valor binario para el tiempo de paquetización del anexo C.

• Aclaración de los principios del mecanismo de comodines y su aplicación en el descriptor de topología.

• Aclaración de las estadísticas y la instrucción Desplazar (move); modificación de las terminaciones mediante los MGC; instrucciones opcionales en una acción; orden de las transacciones; precedencia de la propiedad del modo LocalControl sobre el modo SDP; procesamiento de dígitos; utilización de los símbolos de temporizador del mapa de dígitos en una gama de notación. Utilización de StreamID = 0; retorno de AuditCapabilities para un valor tipo cadena de octetos; negociación de la versión del protocolo.

• Aclaración de que el lote de red puede aplicarse a TDM.

NOTA – Esta Recomendación está formada por el texto de la Rec. UIT-T H.248 con su nuevo número, sus anexos A a E y su apéndice I.

Orígenes

El corrigendum 1 a la Recomendación UIT-T H.248.1 v2 (2002) fue aprobado el 15 de marzo de 2004 por la Comisión de Estudio 16 (2001-2004) del UIT-T por el procedimiento de la Recomendación UIT-T A.8.

PREFACIO

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

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

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

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

NOTA

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

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

PROPIEDAD INTELECTUAL

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

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

ÍNDICE

Recomendación UIT-T H.248.1v2

Protocolo de control de las pasarelas: Versión 2

Corrigendum 1

1) Cláusula 2.1

Modifíquese 2.1 como sigue:

2.1 Referencias normativas

– Recomendación UIT-T H.225.0 (2000), *Protocolos de señalización de llamada y paquetización de trenes de medios para sistemas de comunicación multimedios por paquetes*.

– Recomendación UIT-T H.235 (2000), *Seguridad y criptado para terminales multimedios de la serie H (basados en las Recomendaciones H.323 y H.245)*.

– Recomendación UIT-T H.245 (2001), *Protocolo de control para comunicación multimedios*.

– Recomendación UIT-T H.246 (1998), *Interfuncionamiento de terminales multimedios de la serie H con terminales multimedios de la serie H y terminales vocales/de banda vocal por la RTGC y la RDSI*.

– Recomendación UIT-T H.248.4 (2000), *Protocolo de control de las pasarelas: Transporte por el protocolo de transmisión de control de tren*, más corrigendum 1 (2004).

– Recomendación UIT-T H.248.5 (2000), *Protocolo de control de las pasarelas: Transporte por redes del modo de transferencia asíncrono*.

– Recomendación UIT-T H.248.8 (2002), *Protocolo de control de las pasarelas: Descripción de los códigos de error y de los motivos de cambio de servicio*, más enmienda 1 (2004).

•••••

2) Cláusula 6.2

Modifíquese 6.2 como sigue:

6.2 Terminaciones

•••••

Se pueden aplicar señales a las terminaciones (véase 7.1.11). Se pueden programar terminaciones para detectar eventos, cuya aparición puede provocar la notificación de mensajes al MGC, o una acción de la MG. En una terminación se pueden acumular estadísticas que se comunican al MGC a petición (por medio de la instrucción AuditValue (auditoría de valor), véase 7.2.5) y cuando la terminación deje de existir o regrese a un contexto nulo debido a la instrucción Sustraer (Subtract)~~se substrae del contexto~~.

•••••

6.2.1 Dinámica de las terminaciones

El protocolo puede utilizarse para crear nuevas terminaciones y modificar valores de propiedades de terminaciones existentes. Estas modificaciones incluyen la posibilidad de añadir o suprimir eventos

y/o señales. Las propiedades de terminaciones, y los eventos y señales se describen en las subcláusulas que siguen. Un MGC sólo puede liberar/modificar terminaciones, y los recursos que tales terminaciones representan en el contexto NULL o ~~y~~ que se habían captado, por ejemplo mediante la instrucción Añadir.

6.2.2 TerminationID

Se hace referencia a las terminaciones por un TerminationID, que es un esquema arbitrario elegido por la MG.

Los TerminationID de terminaciones físicas se proporcionan en la pasarela de medios. Se puede optar por que los TerminationID consten de una estructura. Por ejemplo, un TerminationID puede consistir en un grupo de circuitos troncales y una troncal dentro del grupo.

Con los TerminationID puede utilizarse un mecanismo que emplea dos tipos de comodines. Estos dos comodines son ALL y CHOOSE. El primero se utiliza para direccionar múltiples terminaciones de una sola vez, mientras que el segundo se utiliza para indicar a una pasarela de medios que tiene que seleccionar una terminación que satisfaga el TerminationID parcialmente especificado. Esto permite, por ejemplo, que un MGC ordene a la MG que elija un circuito perteneciente a un grupo de troncales.

~~Cuando se utiliza ALL en el TerminationID de una instrucción, el efecto es idéntico al de repetir la instrucción con cada uno de los TerminationID concordantes. La utilización de ALL no direcciona la terminación ROOT. Puesto que cada una de estas instrucciones puede generar una respuesta, el tamaño de la respuesta completa puede ser grande. Si no se necesitan respuestas individuales, se puede pedir una respuesta comodín. En tal caso, se genera una sola respuesta, que contiene la UNION de todas las respuestas individuales que de otro modo se habrían generado, en la que se han suprimido los valores duplicados. Por ejemplo, dada una Termination Ta con propiedades p1 = a, p2 = b y Termination Tb con propiedades p2 = c, p3 = d, una respuesta UNION podría consistir en un TerminationId con comodines y la secuencia de propiedades p1 = a, p2 = b,c y p3 = d. La respuesta comodín puede ser particularmente útil en las instrucciones de auditoría.~~

~~En los anexos A y B se explica la codificación del mecanismo de comodines.~~

6.2.3 Lotes

En los diferentes tipos de pasarelas se pueden implementar terminaciones con características muy diferentes. Las variaciones en las terminaciones se tienen en cuenta en el protocolo permitiendo que las terminaciones tengan propiedades, eventos, señales y estadísticas facultativos implementados por las MG.

•••••

6.2.4 Propiedades de las terminaciones y descriptores

•••••

En el siguiente cuadro se indican todos los posibles descriptores y su utilización. No todos los descriptores son legales como parámetros de entrada o de salida de todas las instrucciones.

Nombre del descriptor	Descripción
Módem	Identifica el tipo de módem, y las propiedades cuando proceda (nota).

•••••

Error	Contiene un código de error y, facultativamente, texto explicativo del error; puede aparecer en respuestas a instrucciones y en peticiones de notificación.
NOTA – El descriptor ModemDescriptor no fue considerado en la Rec. UIT-T H.248.1, Versión ~~1~~2 (03~~5~~/2002).	

6.2.5 Terminación raíz

•••••

3) Cláusula 6.3

Añádase 6.3 como sigue:

6.3 Principios del mecanismo de comodines

En esta cláusula se especifica el comportamiento de las identidades de contexto y terminación del mecanismo de comodines que se aplican a todas las instrucciones. Al procesar estas instrucciones, deben tenerse en cuenta dos tipos de comodines:

1) Comodín de contexto;

2) Comodín de terminación.

Al ejecutar una transacción con contextos comodín y, opcionalmente, terminaciones de comodín, todas las instrucciones de la transacción se ejecutan ordenadamente para un ejemplar concreto ContextID antes de pasar al siguiente ejemplar ContextID. Cuando haya múltiples instrucciones en una transacción, sólo cuando el TerminationID (específico o comodín) especificado en la primera instrucción corresponde a un ejemplar específico de ContextID, se ejecutarán las siguientes instrucciones de la transacción. Si el TerminationID (específico o comodín) de las siguientes instrucciones de la transacción no corresponde a un ejemplar ContextID específico, se devuelve un código de error 431 y se detiene el procesamiento de los siguientes ejemplares ContextID comodín a menos que la instrucción que generó el error se haya marcado como facultativa.

A continuación se indica la ejecución de algunas combinaciones de comodines.

6.3.1 ContextID específico con TerminationID comodín

Cuando el ContextID es específico y se utiliza ALL en el TerminationID de la instrucción, el efecto es idéntico a repetir la instrucción para cada uno de los TerminationID que correspondan. La utilización de ALL no afecta a la terminación ROOT. Cada una de estas instrucciones puede generar una respuesta, por lo que el tamaño de toda la respuesta puede ser grande. Así, si el comodín corresponde con más de un TerminationID en el contexto, se intentan todas las correspondencias posibles y se informa de los resultados obtenidos con cada una de ellas. Si ninguna de las terminaciones a las que hace referencia el TerminationID comodín están en el contexto especificado, se devuelve un código de error 431. No se devuelven los errores de las terminaciones especificadas por el TerminationID comodín que no estén en dicho contexto.

Por ejemplo: Se supone que una pasarela tiene 4 terminaciones: t1/1, t1/2, t2/1 y t2/2, y que el Contexto 1 tiene t1/1 y t2/1 y que el Contexto 2 tiene t1/2 y t2/2.

La instrucción:

Context=1{Command=t1/{Descriptor/s}}*

Retorna:

Context=1{Command=t1/1{Descriptor/s}}

6.3.2 ContextID comodín (ALL) con TerminationID específico

Cuando el ContextID es un comodín (es decir ContextID = ALL) y el TerminationID está totalmente especificado, el efecto es idéntico a una instrucción que indique el contexto no NULL que contiene la terminación especificada. Por consiguiente, debe buscarse el contexto y ejecutarse sólo un ejemplar de la instrucción. No se informan errores para los contextos que no contienen la terminación especificada. Si la terminación no existe en ningún contexto (non-NULL), se devuelve

el error 431. Aunque no se aconseja utilizar esta combinación en lugar de especificar el ContextID, puede resultar útil, por ejemplo, para corregir un estado de conflicto entre la MG y el MGC.

Por ejemplo: Teniendo en cuenta la anterior configuración de pasarela.

La instrucción:

Context={Command=t1/1{Descriptor/s}}*

Retorna:

Context=1{Command=t1/1{Descriptor/s}}

6.3.3 ContextID comodín (ALL) con TerminationID comodín

Cuando tanto el ContextID como el TerminationID son comodines (es decir, ContextID = ALL), el efecto es idéntico a repetir la instrucción para cada TerminationID que corresponda al comodín de cada contexto no NULL que contiene uno o más de esos TerminationID correspondientes. Por consiguiente, si el comodín corresponde con más de un TerminationID en el ejemplar específico del ContextID comodín, se intentan todas las correspondencias posibles y se informa de los resultados de cada una de ellas. No se devuelven los errores de los contextos que no contengan una terminación que corresponda con el TerminationID comodín. No se devuelven los errores de cada terminación especificada en el TerminationID comodín que no estén en un ejemplar específico del ContextID comodín. Si no hay ninguna correspondencia con el ContextID y el TerminationID comodines, se devuelve un código de error 431.

Por ejemplo: Teniendo en cuenta la configuración de pasarela anterior.

La instrucción:

Context={Command=t1/*{Descriptor/s}}*

Retorna:

Context=1{Command=t1/1{Descriptor/s}}

Context=2{Command=t1/2{Descriptor/s}}

Cuando un TerminationID comodín y/o una petición de ContextID comodín contienen múltiples instrucciones, si la primera de ellas no corresponde con el primer ejemplar ContextID y TerminationID, la siguiente instrucción de la petición no se ejecutará para dicho ejemplar.

6.3.4 Respuestas con comodín

Cuando no se necesitan respuestas individuales, puede solicitarse una respuesta comodín. En este caso, se genera una única respuesta que contiene la UNION de todas las respuestas individuales que se hubiesen generado en otro caso, en la que se han suprimido los valores duplicados. Por ejemplo, en una terminación Ta con propiedades p1 = a, p2 = b y una terminación Tb con propiedades p2 = c, p3 = d, una respuesta UNION estará formada por un TerminationID comodín y la secuencia de propiedades p1 = a, p2 = b, c y p3 = d. La respuesta comodín puede resultar especialmente útil para las instrucciones Audit. Si se utiliza una respuesta UNION comodín junto con un contexto comodín, se envía una única respuesta junto con la UNION de todas las terminaciones a que hace referencia el TerminationID. La respuesta contendrá Context = ALL, un TerminationId comodín y la secuencia de propiedades.

En caso de error durante la ejecución de una petición comodín que especifica una respuesta comodín, es necesario aplicar un tratamiento especial para proporcionar información útil sobre los errores limitando el tamaño de la respuesta a una dimensión discreta. Cuando se solicita una respuesta comodín, se ejecutarán todos los ejemplares de la instrucción (como se especificó anteriormente), incluso si uno o más de ellos originan errores, pero no se ejecutarán las instrucciones subsiguientes de la transacción (a menos que se especifique que son facultativos). Se devolverán múltiples respuestas de instrucción para la instrucción que ha encontrado el error. La

primera respuesta de instrucción será una respuesta comodín normal que contenga la UNION de respuestas para las instrucciones que se ejecutaron adecuadamente. Si ninguna de ellas tuvo éxito, la UNION estará vacía. Las respuestas de instrucción adicionales para cada transactionID que no ha tenido éxito se devolverán con el descriptor de error correspondiente.

Por ejemplo

La instrucción:

Context={Command=t1/*{Descriptor/s}}*

Responde a un error:

Context={Command=t1/*{Union response descriptors},*

Command=t1/3{Error=errorcode}}

En los anexos A y B se explica la codificación del mecanismo de comodines.

•••••

4) Cláusula 7, Instrucciones

a) Modifíquese 7.1 como sigue:

7.1.2 Descriptor de módem

El descriptor de módem especifica el tipo de módem y sus parámetros, si procede, requeridos por ejemplo, en conversación con arreglo a H.324 y en conversación mediante texto. El descriptor incluye los siguientes tipos de módems: V.18, V.22, V.22 *bis*, V.32, V.32 *bis*, V.34, V.90, V.91, RDSI síncrona, y permite ampliaciones. Por defecto, en una terminación no hay descriptor de módem.

En la Rec. UIT-T H.248.1, Versión 2~~1~~ (0~~3~~5/2002) y las relaciones más recientes no se consideró el descriptor de módem. Entonces, el descriptor de módem no se incluirá en el contenido transmitido. Si se recibe un descriptor de módem, las condiciones de la implementación determinarán si se trata o no se tiene en cuenta. Debe indicarse el tipo de módem como un atributo de los trenes de datos en LocalDescriptor y RemoteDescriptor.

•••••

7.1.4 Descriptor de medios

El descriptor de medios especifica los parámetros de todos los trenes de medios. Estos parámetros están estructurados en dos descriptores, un descriptor de estado de la terminación (TerminationState), que especifica las propiedades de una terminación que no dependen del tren, y uno o más descriptores de tren, cada uno de los cuales describe un tren de un solo medio.

Un tren se identifica mediante un StreamID. El StreamID estará en la gama de 1 a 65535. El StreamID se utiliza para vincular los trenes en un contexto que corresponden unos con otros. Si existen múltiples trenes una terminación se sincronizará con cada uno de ellos. El descriptor de tren comprende hasta tres descriptores subsidiarios: el descriptor LocalControl, el descriptor local y el descriptor distante. La relación entre estos descriptores es la siguiente:

•••••

7.1.7 Descriptor LocalControl

El descriptor LocalControl contiene la propiedad modo, las propiedades ReserveGroup y ReserveValue y las propiedades de una terminación (definidas en lotes) que son específicas del tren, y son de interés para la MG y el MGC. Los valores de propiedades se pueden especificar como se indica en 7.1.1.

Los valores permitidos para la propiedad modo son sólo enviar, recibir solamente, enviar/recibir, inactivo y conexión en bucle. "Enviar" y "recibir" han de entenderse con respecto al exterior del contexto, de modo que, por ejemplo, un tren fijado a mode = sendOnly no pasa al contexto los medios recibidos. El valor por defecto para la propiedad modo es "Inactive". Las señales y los eventos no son afectados por el modo. La propiedad modo LocalControl tiene precedencia sobre cualquier otro modo especificado en los descriptores Local y Distante.

•••••

7.1.14.2 Temporizadores DigitMap

La colección de dígitos conformes con un DigitMap puede protegerse por tres temporizadores: un temporizador de arranque (T, *start timer*), un temporizador corto (S, *short timer*), y un temporizador largo (L, *long timer*).

1) El temporizador de arranque (T) se utiliza antes de que ~~se haya marcado cualquier número~~ haya cualquier dígito disponible para su procesamiento por referencia al mapa de dígitos. Si el temporizador de arranque se reemplaza por un valor cero, (T = 0), quedará inhabilitado. Esto implica que la MG esperará indefinidamente a recibir las cifras.

•••••

7.1.14.3 Sintaxis de DigitMap

•••••

Además de estos símbolos de eventos, la cadena puede contener los especificadores de temporización entre eventos "S" y "L" y el modificador de duración "Z". "S" y "L" indican respectivamente que la MG debe utilizar el temporizador corto (S) o el temporizador de larga duración (L) para los eventos subsiguientes, haciendo caso omiso de las reglas de temporización descritas anteriormente. Si un especificador de temporización explícito se encuentra realmente en realidad en una secuencia de eventos alternativa, pero no se da ninguno en cualquier otra alternativa candidato, debe utilizarse el valor de temporizador fijado por el especificador de temporización explícito. Si todas las secuencias con controles de temporización explícitos se extraen del conjunto candidato, la temporización regresa a las reglas por defecto dadas anteriormente. Los especificadores S y L se ignorarán si se utilizan dentro de una gama de notación. Por último, si especificadores de temporización divergentes están actuando en secuencias alternativas diferentes, se utilizará el temporizador largo.

"Z" designa un evento de duración larga: colocado al principio del símbolo o símbolos que designan el evento o eventos que satisfacen una posición de dígito dada, indica que esa posición es satisfecha solamente si la duración del evento excede el umbral de la duración larga. Se supone que el valor de este umbral está configurado en la MG, pero puede ser modificado por una especificación en DigitMap, de la misma forma que los temporizadores T, L y S. Si el especificador Z no va seguido de un dígito (0-9 o A-K), la MG rechazará el mapa de dígitos al considerarlo un procedimiento no válido. Cuando se utiliza en una gama de notación, el especificador Z se aplica exclusivamente al dígito inmediatamente siguiente. Cuando se utiliza justo antes de una gama, el modificador Z se aplica a todos los dígitos de la gama (por lo que se requiere que la correspondencia en la gama sea de larga duración).

7.1.14.4 Evento de compleción DigitMap

•••••

7.1.15 Descriptor de estadísticas

El descriptor de estadísticas proporciona información sobre el estado y la utilización de una terminación durante su existencia ~~dentro de un contexto específico~~ (efímera) o cuando está fuera de un contexto NULO (físico). Para cada terminación se mantiene, cuando procede, un conjunto de

estadísticas normalizadas (por ejemplo, el número de octetos enviados y recibidos). Las propiedades estadísticas concretas que son comunicadas para una terminación dada son determinadas por los lotes realizados por la terminación. Por defecto, las estadísticas se comunican cuando la terminación ~~es substraída del contexto~~ deja de existir o se devuelve al contexto NULO de acuerdo con la instrucción Substraer. Para hacerlo de otra forma se puede incluir un AuditDescriptor vacío en la instrucción Substraer. También puede retornar estadísticas la instrucción AuditValue, o cualquier instrucción Añadir/Desplazar/Modificar que utilice el descriptor Audit.

Las estadísticas son acumulativas; el hecho de comunicar estadísticas no las reinicia. Las estadísticas son reiniciadas cuando ~~se substrae una terminación de un contexto~~ una terminación deja de existir o regresa al contexto NULO de acuerdo con la instrucción Substraer.

7.1.16 Descriptor de lotes

•••••

7.1.18 Descriptor de topología

•••••

- (*T1, T2*, aislar) significa que las terminaciones que concuerdan con *T2* no reciben medios desde las terminaciones que concuerdan con *T1*, y viceversa.
- (*T1, T2*, un solo sentido) significa que las terminaciones que concuerdan con *T2* reciben medios de las terminaciones que concuerdan con *T1*, pero no a la inversa. En este caso, está permitida la utilización del comodín ALL de tal modo que haya terminaciones que concuerden con *T1* ~~y~~ o con *T2*, *pero no con ambas*.
- (*T1, T2*, ambos sentidos) significa que las terminaciones que concuerdan con *T2* reciben medios de las terminaciones que concuerdan con *T1*, y viceversa. En este caso se permite utilizar comodines de modo que haya terminaciones que concuerden con *T1* y *T2*. Sin embargo, si hay una terminación que concuerda con ambas, no se introduce una conexión en bucle. Sin embargo, si hay una terminación que concuerda con ambas, no se introduce una conexión en bucle.

•••••

b) Modificar 7.2 como sigue:

7.2.1 Añadir (Add)

La instrucción Añadir añade una terminación a un contexto.

TerminationID

[,MediaDescriptor]

[,ModemDescriptor] (*)

[,MuxDescriptor]

[,EventsDescriptor]

[,SignalsDescriptor]

[,DigitMapDescriptor]

[,ObservedEventsDescriptor]

[,EventBufferDescriptor]

[,StatisticsDescriptor]

[,PackagesDescriptor]

Add(TerminationID

[, MediaDescriptor]

[, ModemDescriptor] (*)

[, MuxDescriptor]
[, EventsDescriptor]
[, EventBufferDescriptor]
[, SignalsDescriptor]
[, DigitMapDescriptor]
[, AuditDescriptor]
)

(*) El descriptor de módem no fue considerado en H.248.1, Versión ~~1~~2 (0~~3~~5/2002).

•••••

7.2.2 Modificar

La instrucción Modificar modifica las propiedades de una terminación.

TerminationID
[,MediaDescriptor]
[,ModemDescriptor] (*)
[,MuxDescriptor]
[,EventsDescriptor]
[,SignalsDescriptor]
[,DigitMapDescriptor]
[,ObservedEventsDescriptor]
[,EventBufferDescriptor]
[,StatisticsDescriptor]
[,PackagesDescriptor]

Modify(TerminationID
[, MediaDescriptor]
[, ModemDescriptor] (*)
[, MuxDescriptor]
[, EventsDescriptor]
[, EventBufferDescriptor]
[, SignalsDescriptor]
[, DigitMapDescriptor]
[, AuditDescriptor]
)

(*) El descriptor de módem no fue considerado en H.248.1, Versión ~~1~~2 (0~~3~~5/2002).

•••••

7.2.3 Substraer

La instrucción Substraer desconecta una terminación de su contexto y retorna estadísticas sobre la participación de la terminación en el contexto.

TerminationID
[,MediaDescriptor]
[,ModemDescriptor] (*)

[,MuxDescriptor]

[,EventsDescriptor]

[,SignalsDescriptor]

[,DigitMapDescriptor]

[,ObservedEventsDescriptor]

[,EventBufferDescriptor]

[,StatisticsDescriptor]

[,PackagesDescriptor]

Subtract(TerminationID

[, AuditDescriptor]

)

(*) El descriptor de módem no fue considerado en H.248.1, Versión ~~1~~2 (0~~3~~5/2002).

•••••

7.2.4 Desplazar

La instrucción Desplazar desplaza una terminación de su contexto actual a otro contexto, en una operación atómica. La instrucción Desplazar es la única instrucción que hace referencia a una terminación en un contexto diferente del contexto al que se aplica la instrucción. La instrucción Desplazar no se utilizará para desplazar terminaciones al contexto nulo, o desde dicho contexto.

TerminationID

[,MediaDescriptor]

[,ModemDescriptor] (*)

[,MuxDescriptor]

[,EventsDescriptor]

[,SignalsDescriptor]

[,DigitMapDescriptor]

[,ObservedEventsDescriptor]

[,EventBufferDescriptor]

[,StatisticsDescriptor]

[,PackagesDescriptor]

Move(TerminationID

[, MediaDescriptor]

[, ModemDescriptor] (*)

[, MuxDescriptor]

[, EventsDescriptor]

[, EventBufferDescriptor]

[, SignalsDescriptor]

[, DigitMapDescriptor]

[, AuditDescriptor]

)

(*) El descriptor de módem no fue considerado en H.248.1, Versión ~~1~~2 (0~~3~~5/2002).

•••••

7.2.5 AuditValue

La instrucción AuditValue retorna los valores actuales de propiedades, eventos, señales y estadísticas asociadas con terminaciones. AuditValue puede pedir el contenido de un descriptor o sólo de una propiedad, un evento, una señal o estadísticas.

TerminationID

[,MediaDescriptor]

[,ModemDescriptor] (*)

[,MuxDescriptor]

[,EventsDescriptor]

[,SignalsDescriptor]

[,DigitMapDescriptor]

[,ObservedEventsDescriptor]

[,EventBufferDescriptor]

[,StatisticsDescriptor]

[,PackagesDescriptor]

AuditValue(TerminationID,

AuditDescriptor

)

(*) El descriptor de módem no fue considerado en H.248.1, Versión ~~1~~2 (0~~3~~5/2002).

•••••

7.2.6 AuditCapabilities

La instrucción AuditCapabilities retorna los posibles valores de propiedades, eventos, señales y estadísticas asociados con terminaciones. Puede pedirse AuditCapabilities para el contenido de un descriptor o para una determinada propiedad, evento, señal o estadísticas.

TerminationID

[,MediaDescriptor]

[,ModemDescriptor](*)

[,MuxDescriptor]

[,EventsDescriptor]

[,SignalsDescriptor]

[,ObservedEventsDescriptor]

[,EventBufferDescriptor]

[,StatisticsDescriptor]

AuditCapabilities(TerminationID,

AuditDescriptor)

(*) El descriptor de módem no se consideró en H.248.1, Versión ~~1~~2 (0~~3~~5/2002).

•••••

Si se solicita una respuesta con comodín, solamente se genera un retorno de instrucción cuyo contenido está formado por la unión de los valores de todas las terminaciones concordantes con el comodín. Este convenio puede reducir el volumen de los datos requeridos para efectuar una auditoría de un grupo de terminaciones.

Cuando se hace la auditoría de una propiedad, una señal, un evento o estadísticas, AuditCapabilities hace que el sistema retorne las propiedades, las señales, los eventos y las estadísticas apropiadas con las capacidades de la terminación.

La interpretación de las capacidades que son solicitadas para los diversos valores de ContextID y TerminationID es la misma que en AuditValue.

En el caso de los valores de propiedades y parámetros de tipo cadena de caracteres, carácter o cadena de octetos, la MG devolverá un valor vacío. Para la codificación de texto, las cadenas de caracteres y los caracteres devuelven una construcción quotedString vacía, mientras que las cadenas de octetos devuelven una respuesta NULL (0x00). Este comportamiento puede ser anulado por la definición del lote.

EventsDescriptor retorna la lista de posibles eventos en la terminación junto con la lista de todos los valores posibles de los parámetros de EventsDescriptor. EventBufferDescriptor retorna la misma información que EventsDescriptor. SignalsDescriptor retorna la lista de las posibles señales que podrían aplicarse a la terminación junto con la lista de todos los posibles valores de los parámetros de señales. StatisticsDescriptor retorna los valores de las estadísticas que se llevan en la terminación. ObservedEventsDescriptor retorna los nombres de los eventos activos en la terminación. DigitMap y lotes no legales en AuditCapability.

•••••

5) Cláusula 8

Modifíquese la cláusula 8 como sigue:

8 Transacciones

•••••

Las transacciones son TransactionRequests. Las respuestas correspondientes a una TransactionRequest se reciben en una sola contestación, posiblemente precedida por varios mensajes TransactionPending (véase 8.2.3).

Las transacciones garantizan el procesamiento ordenado de las instrucciones, se ejecutan en la misma secuencia. La ordenación de las transacciones NO está garantizada; las transacciones pueden ejecutarse en cualquier orden, o simultáneamente~~.~~; no obstante, las respuestas a las transacciones deben ejecutarse antes de las peticiones de las transacciones, cuando ambas están contenidas en un mensaje.

Al primer fallo de una instrucción en una transacción, se detiene el procesamiento de las instrucciones restantes en esa transacción. Si una instrucción contiene un TerminationID con comodines, se trata de ejecutar la instrucción con cada uno de los TerminationID que concuerdan con el comodín. En TransactionReply se incluye una respuesta para cada TerminationID concordante, incluso si una o más ejemplares generaron un error. Si cualquier TerminationID concordante con un comodín produce un error cuando se ejecuta, no se intentará ninguna de las instrucciones que siguen a la instrucción con el comodín.

•••••

8.2.1 TransactionRequest

La TransactionRequest la invoca el emisor y se realiza una transacción para cada invocación de petición. Una petición contiene una o más acciones, cada una de las cuales especifica el contexto deseado~~ y una o más instrucciones por contexto~~.

TransactionRequest(TransactionId {

ContextID {Command … Command},

. . .

ContextID {Command … Command } })

El parámetro TransactionID especificará un valor para ulterior correlación con la respuesta TransactionReply o TransactionPending del receptor.

•••••

6) Cláusula 11.3

Modifíquese 11.3 como sigue:

11.3 Negociación de la versión de protocolo

La primera instrucción ServiceChange procedente de una MG contendrá el número de versión del protocolo soportado por la MG en el parámetro ServiceChangeVersion. Después de la recepción de este mensaje, y si el MGC soporta solamente una versión inferior, el MGC enviará entonces una ServiceChangeReply con la versión inferior y más tarde todos los mensajes entre la MG y el MGC se conformarán a la versión inferior del protocolo. Si la MG no puede cumplimentar lo anterior y ha establecido una conexión de transporte con el MGC, debe cerrar dicha conexión. En cualquier caso, debe rechazar todas las peticiones subsiguientes procedentes del MGC con error 406 (Versión no soportada).

Si el MGC únicamente soporta una versión superior a la de la MG, rechazará la asociación mediante el error 406 (versión no soportada).

Si el MGC soporta la versión señalada por la MG, adaptará a dicha versión los mensajes subsiguientes. En este caso, el MGC puede devolver facultativamente una versión en el ServiceChangeReply. ~~Si el MGC soporta una versión superior que la versión de la MG pero no puede soportar la versión inferior propuesta por la MG, enviará una ServiceChangeReply con la versión inferior y más tarde todos los mensajes entre la MG y el MGC se conformarán a la versión inferior del protocolo. Si el MGC no puede cumplimentar lo anterior, rechazará la asociación mediante el error 406 (Versión no soportada).~~

La negociación de la versión del protocolo también puede ocurrir en los ServiceChanges "traspaso" y "cambio-en-caso-de-fallo".

•••••

7) Cláusula 12

Modifíquese la cláusula 12 como sigue:

12.1.5 Estadísticas

Estadísticas definidas por el lote, que especifica:

Nombre de la estadística: solamente descriptivo

StatisticID: es un identificador

StatisticID se utiliza en un StatisticsDescriptor

Descripción:

Tipo: uno de los siguientes:

Booleano

Cadena de caracteres: cadena UTF-8

Cadena de octetos: Varios octetos. Para la codificación, véase el anexo A y B.3.

Entero: entero de 4 bytes con signo

Doble: entero de 8 bytes con signo

Carácter: codificación UTF-8 Unicode de una sola letra. Podría tener más de un octeto.

Enumeración: un valor de una lista de valores únicos posibles (véase 12.3)

Sublista: una lista de varios valores de una lista. El tipo de sublista SE ESPECIFICARÁ también. El tipo se elegirá entre los tipos especificados en esta cláusula (salvo de la sublista). Por ejemplo, tipo: sublista de enumeración. La codificación de las sublistas se especifica en el anexo A y B.3.

Valores posibles:

Un lote debe indicar la unidad de medida, por ejemplo, milisegundos, paquetes, aquí o dentro del tipo señalado anteriormente, así como cualquier restricción de la gama.

~~Unidades: unidad de medida, por ejemplo milisegundos, paquetes~~

12.1.6 Procedimientos

•••••

12.5 Registro de lotes

Un lote puede registrarse en la IANA por razones de interoperabilidad. Para las consideraciones relativas a la IANA véase la cláusula ~~13~~14.

•••••

8) Anexo B Codificación textual del protocolo

Modifíquese el anexo B como sigue:

•••••

B.2 Especificación ABNF

La sintaxis de protocolo se presenta en ABNF de acuerdo con RFC 2234.

NOTA 1 – Esta especificación de sintaxis no impone todas las restricciones para valores o introducción de elementos. Algunas restricciones adicionales se establecen en comentarios y otras restricciones aparecen en el texto de esta Recomendación. Estas restricciones adicionales forman parte del protocolo aunque no vengan impuestas por esta Recomendación.

NOTA 2 – La sintaxis es independiente del contenido. Por ejemplo, "Add" puede ser el AddToken o un NAME que depende del contexto en que aparezca.

Todo lo escrito en ABNF y en codificación de texto es insensible al hecho de que utilicen mayúsculas o minúsculas. Esto incluye los TerminationID, los identificadores de mapa de dígitos, etc. SPD es sensible a la utilización de mayúsculas o minúsculas según RFC 2327.

```
; NOTE - The ABNF in this section uses the VALUE construct (or lists of
; VALUE constructs) to encode various package element values (properties,
; signal parameters, etc.). The types of these values vary and are
; specified in the relevant package definition. Several such types are
; described in 12.2.
;
; The ABNF specification for VALUE allows a quotedString form or a
; collection of SafeChars. The encoding of package element values into
; ABNF VALUES is specified below. If a type's encoding allows characters
; other than SafeChars, the quotedString form MUST be used for all values
; of that type, even for specific values that consist only of SafeChars.
;
; String: A string MUST use the quotedString form of VALUE and can
```

```
; contain anything allowable in the quotedString form.
;
; Integer, Double, and Unsigned Integer:  Decimal values can be encoded
; using characters 0-9.  Hexadecimal values must be prefixed with '0x'
; and can use characters 0-9,a-f,A-F.  An octal format is not supported.
; Negative integers start with '-' and MUST be Decimal.  The SafeChar
; form of VALUE MUST be used.
;
; Character: A UTF-8 encoding of a single letter surrounded by double
; quotes.
;
; Enumeration: An enumeration MUST use the SafeChar form of VALUE
; and can contain anything allowable in the SafeChar form.
;
; Boolean: Boolean values are encoded as "on" and "off" and are
; case insensitive. The SafeChar form of VALUE MUST be used.
;
; Future types: Any defined types MUST fit within
; the ABNF specification of VALUE. Specifically, if a type's encoding
; allows characters other than SafeChars, the quotedString form MUST
; be used for all values of that type, even for specific values that
; consist only of SafeChars.
;
; Note that there is no way to use the double quote character within
; a value.
;
; Note that SDP disallows whitespace at the beginning of a line, Megaco
; ABNF allows whitespace before the beginning of the SDP in the
; Local/Remote descriptor. Parsers should accept whitespace between the
; LBRKT following the Local/Remote token and the beginning of the SDP.

megacoMessage        = LWSP [authenticationHeader SEP ] message

authenticationHeader = AuthToken EQUAL SecurityParmIndex COLON
                       SequenceNum COLON AuthData

SecurityParmIndex    = "0x" 8(HEXDIG)
SequenceNum          = "0x" 8(HEXDIG)
AuthData             = "0x" 24*64(HEXDIG)

message            = MegacopToken SLASH Version SEP mId SEP messageBody
; The version of the protocol defined here is equal to 2.

messageBody          = ( errorDescriptor / transactionList )

transactionList      = 1*( transactionRequest / transactionReply /
                       transactionPending / transactionResponseAck )
;Use of response acks is dependent on underlying transport

transactionPending   = PendingToken EQUAL TransactionID LBRKT RBRKT

transactionResponseAck = ResponseAckToken LBRKT transactionAck
                         *(COMMA transactionAck) RBRKT
transactionAck = TransactionID / (TransactionID "-" TransactionID)

transactionRequest   = TransToken EQUAL TransactionID LBRKT
                       actionRequest *(COMMA actionRequest) RBRKT

actionRequest        = CtxToken EQUAL ContextID LBRKT ((
                       contextRequest [COMMA  commandRequestList])
                       / commandRequestList) RBRKT

contextRequest    = ((contextProperties [COMMA contextAudit])
                  / contextAudit)
```

```
contextProperties    = contextProperty *(COMMA contextProperty)

; at-most-once
; EmergencyOff to be used in MG to MGC direction only in H.248.1 V1 and V2
; EmergencyToken or EmergencyOffToken, but not both
contextProperty      = (topologyDescriptor / priority / EmergencyToken /
                        EmergencyOffToken)

contextAudit     = ContextAuditToken LBRKT
                       contextAuditProperties *(COMMA
                       contextAuditProperties) RBRKT

; at-most-once
contextAuditProperties = ( TopologyToken / EmergencyToken /
                           PriorityToken )

; "O-" indicates an optional command
; "W-" indicates a wildcarded response to a command
commandRequestList= ["O-"] ["W-"] commandRequest *
(COMMA ["O-"] ["W-"]commandRequest)

commandRequest       = ( ammRequest / subtractRequest / auditRequest /
                         notifyRequest / serviceChangeRequest)

transactionReply        = ReplyToken EQUAL TransactionID LBRKT
                                  [ ImmAckRequiredToken COMMA]
                          ( errorDescriptor / actionReplyList ) RBRKT

actionReplyList      = actionReply *(COMMA actionReply )

actionReply          = CtxToken EQUAL ContextID LBRKT
                       ( errorDescriptor / commandReply /
                       (commandReply COMMA errorDescriptor) ) RBRKT

commandReply       = (( contextProperties [COMMA commandReplyList] ) /
                         commandReplyList )


commandReplyList     = commandReplys *(COMMA commandReplys )

commandReplys        = (serviceChangeReply / auditReply / ammsReply /
                         notifyReply )

;Add Move and Modify have the same request parameters
ammRequest           = (AddToken / MoveToken / ModifyToken ) EQUAL
                       TerminationID [LBRKT ammParameter *(COMMA
                       ammParameter) RBRKT]

;at-most-once
ammParameter         = (mediaDescriptor / modemDescriptor /
                         muxDescriptor / eventsDescriptor /
                         signalsDescriptor / digitMapDescriptor /
                         eventBufferDescriptor / auditDescriptor)

ammsReply            = (AddToken / MoveToken / ModifyToken /
                         SubtractToken ) EQUAL TerminationID [ LBRKT
                         terminationAudit RBRKT ]

subtractRequest      =  SubtractToken EQUAL TerminationID
                         [ LBRKT auditDescriptor RBRKT]

auditRequest         =  (AuditValueToken / AuditCapToken ) EQUAL
                        TerminationID LBRKT auditDescriptor RBRKT
```

```
auditReply           = (AuditValueToken / AuditCapToken )
                       ( contextTerminationAudit  / auditOther)

auditOther           = EQUAL TerminationID [LBRKT
                       terminationAudit RBRKT]

terminationAudit   = auditReturnParameter *(COMMA auditReturnParameter)

contextTerminationAudit = EQUAL CtxToken ( terminationIDList /
                        LBRKT errorDescriptor RBRKT )

auditReturnParameter = (mediaDescriptor / modemDescriptor /
                        muxDescriptor / eventsDescriptor /
                        signalsDescriptor / digitMapDescriptor /
                        observedEventsDescriptor / eventBufferDescriptor /
                        statisticsDescriptor / packagesDescriptor /
                        errorDescriptor / auditReturnItem)

auditReturnItem      = (MuxToken / ModemToken / MediaToken /
                        DigitMapToken / StatsToken /
                        ObservedEventsToken / PackagesToken)

auditDescriptor      = AuditToken LBRKT [ auditItem
                       *(COMMA auditItem) ] RBRKT

notifyRequest        = NotifyToken EQUAL TerminationID
                       LBRKT ( observedEventsDescriptor
                             [ COMMA errorDescriptor ] ) RBRKT

notifyReply          = NotifyToken EQUAL TerminationID
                       [ LBRKT errorDescriptor RBRKT ]

serviceChangeRequest = ServiceChangeToken EQUAL TerminationID
                       LBRKT serviceChangeDescriptor RBRKT

serviceChangeReply   = ServiceChangeToken EQUAL TerminationID
                       [LBRKT (errorDescriptor /
                       serviceChangeReplyDescriptor) RBRKT]

errorDescriptor   = ErrorToken EQUAL ErrorCode
                    LBRKT [quotedString] RBRKT

ErrorCode            = 1*4(DIGIT) ; could be extended

TransactionID        = UINT32

mId                  = (( domainAddress / domainName )
                       [":" portNumber]) / mtpAddress / deviceName

; ABNF allows two or more consecutive "." although it is meaningless
; in a domain name.
domainName           = "<" (ALPHA / DIGIT) *63(ALPHA / DIGIT / "-" /
                       ".") ">"
deviceName           = pathNAME

;The values 0x0, 0xFFFFFFFE and 0xFFFFFFFF are reserved.
;'-' is used for NULL context.
ContextID            = (UINT32 / "*" / "-" / "$")

domainAddress        = "[" (IPv4address / IPv6address) "]"
;RFC 2373 contains the definition of IP6Addresses.
IPv6address          = hexpart [ ":" IPv4address ]
IPv4address          = V4hex DOT V4hex DOT V4hex DOT V4hex
```

```
V4hex                = 1*3(DIGIT) ; "0".."255"
; this production, while occurring in RFC 2373, is not referenced
; IPv6prefix           = hexpart SLASH 1*2DIGIT
hexpart             = hexseq "::" [ hexseq ] / "::" [ hexseq ] / hexseq
hexseq               = hex4 *( ":" hex4)
hex4                 = 1*4HEXDIG

portNumber           = UINT16

; Addressing structure of mtpAddress:
; 25 - 15            0
;    |  PC        | NI |
;    24 - 14 bits    2 bits
; Note: 14 bits are defined for international use.
; Two national options exist where the point code is 16 or 24 bits.
; To octet align the mtpAddress the MSBs shall be encoded as 0s.
; An octet shall be represented by 2 hex digits.
mtpAddress           = MTPToken LBRKT 4*8 (HEXDIG) RBRKT

terminationIDList    = LBRKT TerminationID *(COMMA TerminationID) RBRKT

; Total length of pathNAME must not exceed 64 chars.
pathNAME             = ["*"] NAME *("/" / "*"/ ALPHA / DIGIT /"_" / "$" )
                       ["@" pathDomainName ]

; ABNF allows two or more consecutive "." although it is meaningless
; in a path domain name.
pathDomainName       = (ALPHA / DIGIT / "*" )
                       *63(ALPHA / DIGIT / "-" / "*" / ".")

TerminationID        = "ROOT" / pathNAME / "$" / "*"

mediaDescriptor   = MediaToken LBRKT mediaParm *(COMMA mediaParm) RBRKT

; at-most one terminationStateDescriptor
; and either streamParm(s) or streamDescriptor(s) but not both
mediaParm            = (streamParm / streamDescriptor /
                        terminationStateDescriptor)

; at-most-once per item
streamParm           = ( localDescriptor / remoteDescriptor /
                        localControlDescriptor )

streamDescriptor     = StreamToken EQUAL StreamID LBRKT streamParm
                       *(COMMA streamParm) RBRKT

localControlDescriptor = LocalControlToken LBRKT localParm
                         *(COMMA localParm) RBRKT

; at-most-once per item except for propertyParm
localParm            = ( streamMode / propertyParm / reservedValueMode
     / reservedGroupMode )

reservedValueMode     = ReservedValueToken EQUAL ( "ON" / "OFF" )
reservedGroupMode     = ReservedGroupToken EQUAL ( "ON" / "OFF" )

streamMode           = ModeToken EQUAL streamModes

streamModes          = (SendonlyToken / RecvonlyToken / SendrecvToken /
                       InactiveToken / LoopbackToken )

propertyParm         = pkgdName parmValue
parmValue            = (EQUAL alternativeValue/ INEQUAL VALUE)
alternativeValue     = ( VALUE
```

```
                    / LSBRKT VALUE *(COMMA VALUE) RSBRKT
                          ; sublist (i.e. A AND B AND ...)
                    / LBRKT VALUE *(COMMA VALUE) RBRKT
                     ; alternatives (i.e. A OR B OR ...)

                    /  LSBRKT VALUE COLON VALUE RSBRKT )
                     ; range


INEQUAL               = LWSP (">" / "<" / "#" ) LWSP ; '#' means "not equal"
LSBRKT                = LWSP "[" LWSP
RSBRKT                = LWSP "]" LWSP

; Note - The octet zero is not among the permitted characters in octet
; string. As the current definition is limited to SDP, and a zero octet
; would not be a legal character in SDP, this is not a concern.
localDescriptor       = LocalToken LBRKT octetString RBRKT

remoteDescriptor      = RemoteToken LBRKT octetString RBRKT

eventBufferDescriptor= EventBufferToken [ LBRKT eventSpec
                        *( COMMA eventSpec) RBRKT ]

eventSpec      = pkgdName [ LBRKT eventSpecParameter
                   *(COMMA eventSpecParameter) RBRKT ]
eventSpecParameter    = (eventStream / eventOther)

eventBufferControl      = BufferToken EQUAL ( "OFF" / LockStepToken )

terminationStateDescriptor = TerminationStateToken LBRKT
            terminationStateParm *( COMMA terminationStateParm ) RBRKT

; at-most-once per item except for propertyParm
terminationStateParm =(propertyParm / serviceStates / eventBufferControl )

serviceStates         = ServiceStatesToken EQUAL ( TestToken /
                        OutOfSvcToken / InSvcToken )

muxDescriptor         = MuxToken EQUAL MuxType  terminationIDList

MuxType               = ( H221Token / H223Token / H226Token / V76Token
                           / extensionParameter / Nx64kToken )

StreamID              = UINT16
pkgdName              = (PackageName SLASH ItemID) ;specific item
                              / (PackageName SLASH "*") ;all items in package
                              / ("*" SLASH "*") ; all items supported by the MG
PackageName           = NAME
ItemID                = NAME

eventsDescriptor      = EventsToken [ EQUAL RequestID LBRKT
                        requestedEvent *( COMMA requestedEvent ) RBRKT ]

requestedEvent        = pkgdName [ LBRKT eventParameter
                        *( COMMA eventParameter ) RBRKT ]

; at-most-once each of KeepActiveToken , eventDM and eventStream
; at most one of either embedWithSig or embedNoSig but not both
; KeepActiveToken and embedWithSig must not both be present
eventParameter        = ( embedWithSig / embedNoSig / KeepActiveToken
                        /eventDM / eventStream / eventOther )

embedWithSig          = EmbedToken LBRKT signalsDescriptor
                        [COMMA embedFirst ] RBRKT
```

```
embedNoSig       = EmbedToken LBRKT embedFirst RBRKT

; at-most-once of each
embedFirst       = EventsToken [ EQUAL RequestID LBRKT
                secondRequestedEvent *(COMMA secondRequestedEvent) RBRKT ]

secondRequestedEvent = pkgdName [ LBRKT secondEventParameter
                       *( COMMA secondEventParameter ) RBRKT ]

; at-most-once each of embedSig , KeepActiveToken, eventDM or
; eventStream
; KeepActiveToken and embedSig must not both be present
secondEventParameter = ( embedSig / KeepActiveToken / eventDM /
                         eventStream / eventOther )

embedSig  = EmbedToken LBRKT signalsDescriptor RBRKT

eventStream          = StreamToken EQUAL StreamID

eventOther           = eventParameterName parmValue

eventParameterName   = NAME

eventDM              = DigitMapToken EQUAL(( digitMapName ) /
                       (LBRKT digitMapValue RBRKT ))

signalsDescriptor    = SignalsToken [ LBRKT signalParm
                       *(COMMA signalParm) RBRKT ]

signalParm           = signalList / signalRequest

signalRequest        = signalName [ LBRKT sigParameter
                       *(COMMA sigParameter) RBRKT ]

signalList           = SignalListToken EQUAL signalListId LBRKT
                       signalListParm *(COMMA signalListParm) RBRKT

signalListId         = UINT16

;exactly once signalType, at most once duration and every signal
;parameter
signalListParm       = signalRequest

signalName           = pkgdName
;at-most-once sigStream, at-most-once sigSignalType,
;at-most-once sigDuration, every signalParameterName at most once
sigParameter       = sigStream / sigSignalType / sigDuration / sigOther
                        / notifyCompletion / KeepActiveToken
sigStream            = StreamToken EQUAL StreamID
sigOther             = sigParameterName parmValue
sigParameterName     = NAME
sigSignalType        = SignalTypeToken EQUAL signalType
signalType           = (OnOffToken / TimeOutToken / BriefToken)
sigDuration          = DurationToken EQUAL UINT16
notifyCompletion     = NotifyCompletionToken EQUAL (LBRKT
            notificationReason *(COMMA notificationReason) RBRKT)

notificationReason   = ( TimeOutToken / InterruptByEventToken
                       / InterruptByNewSignalsDescrToken
                       / OtherReasonToken )
observedEventsDescriptor = ObservedEventsToken EQUAL RequestID
                  LBRKT observedEvent *(COMMA observedEvent) RBRKT

; time per event, because it might be buffered
```

```
observedEvent        = [ TimeStamp LWSP COLON] LWSP
                       pkgdName [ LBRKT observedEventParameter
                       *(COMMA observedEventParameter) RBRKT ]

; at-most-once eventStream, every eventParameterName at most once
observedEventParameter = eventStream / eventOther

; For an AuditCapReply with all events, the RequestID should be ALL.
RequestID            = ( UINT32 / "*" )

modemDescriptor      = ModemToken (( EQUAL modemType) /
                       (LSBRKT modemType *(COMMA modemType) RSBRKT))
                       [ LBRKT propertyParm
                      *(COMMA propertyParm) RBRKT ]

; at-most-once except for extensionParameter
modemType            = (V32bisToken / V22bisToken / V18Token /
                        V22Token / V32Token / V34Token / V90Token /
                        V91Token / SynchISDNToken / extensionParameter)

digitMapDescriptor   = DigitMapToken EQUAL
                       ( ( LBRKT digitMapValue RBRKT )
                  / (digitMapName [ LBRKT digitMapValue RBRKT ]) )
digitMapName         = NAME
digitMapValue        = ["T" COLON Timer COMMA] ["S" COLON Timer COMMA]
                       ["L" COLON Timer COMMA] ["Z" COLON Timer COMMA]
                       digitMap
Timer                = 1*2DIGIT
; Units are seconds for T, S, and L timers, and hundreds of
; milliseconds for Z timer. Thus T, S, and L range from 1 to 99
; seconds and Z from 100 ms to 9.9 s
digitMap = (digitString / LWSP "(" LWSP digitStringList LWSP ")" LWSP)
digitStringList      = digitString *( LWSP "|" LWSP digitString )
digitString          = 1*(digitStringElement)
digitStringElement   = digitPosition [DOT]
digitPosition        = digitMapLetter / digitMapRange
digitMapRange        = ("x" / (LWSP "[" LWSP digitLetter LWSP "]" LWSP))
digitLetter          = *((DIGIT "-" DIGIT ) / digitMapLetter)
digitMapLetter       = DIGIT   ;Basic event symbols
                       / %x41-4B / %x61-6B ; a-k, A-K
                       / "L" / "S" / "T" ;Inter-event timers
                       ; (long, short, start)
                       / "Z"    ;Long duration modifier

; at-most-once, and DigitMapToken and PackagesToken are not allowed
; in AuditCapabilities command
auditItem            = (~~MuxToken / ModemToken / MediaToken /~~ auditReturnItem /
                        SignalsToken /
                        EventBufferToken /
                        ~~DigitMapToken / StatsToken /~~ EventsToken /
                        ~~ObservedEventsToken / PackagesToken ) /~~
                       indAudterminationAudit)

indAudterminationAudit   = indAudauditReturnParameter
                           *(COMMA indAudauditReturnParameter)

indAudauditReturnParameter = (indAudmediaDescriptor / /
                              indAudeventsDescriptor /
                              indAudsignalsDescriptor /
                              indAuddigitMapDescriptor /
                              indAudeventBufferDescriptor /
                              indAudstatisticsDescriptor /
                              indAudpackagesDescriptor)
```

```
indAudmediaDescriptor   = MediaToken LBRKT indAudmediaParm RBRKT

; at-most-once per item
; and either streamParm or streamDescriptor but not both
indAudmediaParm            = (indAudstreamParm / indAudstreamDescriptor /
                                  indAudterminationStateDescriptor)

; at-most-once
indAudstreamParm           = ( indAudlocalControlDescriptor )
; SDP too complex to pull out individual pieces for audit,
; hence no individual audit for Local and Remote

indAudstreamDescriptor     = StreamToken EQUAL StreamID
                                   LBRKT indAudstreamParm RBRKT

indAudlocalControlDescriptor = LocalControlToken LBRKT indAudlocalParm RBRKT

; at-most-once per item
indAudlocalParm            = ( ModeToken / pkgdName /
                               ReservedValueToken /
                               ReservedGroupToken )


indAudterminationStateDescriptor = TerminationStateToken LBRKT
                             indAudterminationStateParm RBRKT

; at-most-once per item
indAudterminationStateParm =(pkgdName / ServiceStatesToken / BufferToken)

indAudeventBufferDescriptor = EventBufferToken LBRKT indAudeventSpec RBRKT

indAudeventSpec         = pkgdName [ LBRKT indAudeventSpecParameter RBRKT ]
indAudeventSpecParameter   = (eventStream / eventParameterName)

indAudeventsDescriptor     = EventsToken EQUAL RequestID LBRKT
                            indAudrequestedEvent RBRKT

indAudrequestedEvent       = pkgdName

indAudsignalsDescriptor    = SignalsToken LBRKT [ indAudsignalParm ] RBRKT

indAudsignalParm           = indAudsignalList / indAudsignalRequest

indAudsignalRequest        = signalName
indAudsignalList           = SignalListToken EQUAL signalListId LBRKT
                                  indAudsignalListParm RBRKT

indAudsignalListParm    = indAudsignalRequest

indAuddigitMapDescriptor   = DigitMapToken EQUAL (digitMapName )

indAudstatisticsDescriptor = StatsToken LBRKT pkgdName RBRKT

indAudpackagesDescriptor   = PackagesToken LBRKT packagesItem RBRKT


serviceChangeDescriptor = ServicesToken LBRKT serviceChangeParm
                         *(COMMA serviceChangeParm) RBRKT

; each parameter at-most-once, except auditItem
; at most one of either serviceChangeAddress or serviceChangeMgcId but
; not both
; serviceChangeMethod and serviceChangeReason are REQUIRED
```

```
serviceChangeParm    = (serviceChangeMethod / serviceChangeReason /
                        serviceChangeDelay / serviceChangeAddress /
                        serviceChangeProfile / extension / TimeStamp /
                        serviceChangeMgcId / serviceChangeVersion /
                        auditItem)

serviceChangeReplyDescriptor = ServicesToken LBRKT
                        servChgReplyParm *(COMMA servChgReplyParm) RBRKT

; at-most-once. Version is REQUIRED on first ServiceChange response
; at most one of either serviceChangeAddress or serviceChangeMgcId but
; not both
servChgReplyParm     = (serviceChangeAddress / serviceChangeMgcId /
                        serviceChangeProfile / serviceChangeVersion /
                        TimeStamp)
serviceChangeMethod  = MethodToken EQUAL (FailoverToken /
                        ForcedToken / GracefulToken / RestartToken /
                        DisconnectedToken / HandOffToken /
                        extensionParameter)
; A serviceChangeReason consists of a numeric reason code
; and an optional text description.
; A serviceChangeReason MUST be encoded using the quotedString
; form of VALUE.
; The quotedString SHALL contain a decimal reason code,
; optionally followed by a single space character and a
; textual description string.


serviceChangeReason  = ReasonToken  EQUAL VALUE
serviceChangeDelay   = DelayToken   EQUAL UINT32
serviceChangeAddress = ServiceChangeAddressToken EQUAL ( mId /
                        portNumber )
serviceChangeMgcId   = MgcIdToken   EQUAL mId
serviceChangeProfile = ProfileToken EQUAL NAME SLASH Version
serviceChangeVersion = VersionToken EQUAL Version
extension            = extensionParameter parmValue

packagesDescriptor   = PackagesToken LBRKT packagesItem
                        *(COMMA packagesItem) RBRKT

Version              = 1*2(DIGIT)
packagesItem         = NAME "-" UINT16

TimeStamp            = Date "T" Time ; per ISO 8601:1988
; Date = yyyymmdd
Date                 = 8(DIGIT)
; Time = hhmmssss
Time                 = 8(DIGIT)
statisticsDescriptor = StatsToken LBRKT statisticsParameter
                       *(COMMA statisticsParameter ) RBRKT

;at-most-once per item
statisticsParameter  = pkgdName [EQUAL VALUE]

topologyDescriptor   = TopologyToken LBRKT topologyTriple
                        *(COMMA topologyTriple) RBRKT
topologyTriple       = terminationA COMMA
                        terminationB COMMA topologyDirection
                        [COMMA eventStream ]
terminationA         = TerminationID
terminationB         = TerminationID
topologyDirection    = BothwayToken / IsolateToken / OnewayToken

priority             = PriorityToken EQUAL UINT16
```

```
extensionParameter   = "X"  ("-" / "+") 1*6(ALPHA / DIGIT)

; octetString is used to describe SDP defined in RFC 2327.
; Caution should be taken if CRLF in RFC 2327 is used.
; To be safe, use EOL in this ABNF.
; Whenever "}" appears in SDP, it is escaped by "\", e.g. "\}"
octetString          = *(nonEscapeChar)
nonEscapeChar        = ( "\}" / %x01-7C / %x7E-FF )
; Note - The double-quote character is not allowed in quotedString.
quotedString         = DQUOTE *(SafeChar / RestChar/ WSP) DQUOTE

UINT16               = 1*5(DIGIT)  ; %x0-FFFF
UINT32               = 1*10(DIGIT) ; %x0-FFFFFFFF

NAME                 = ALPHA *63(ALPHA / DIGIT / "_" )
VALUE                = quotedString / 1*(SafeChar)
SafeChar             = DIGIT / ALPHA / "+" / "-" / "&" /
                       "!" / "_" / "/" / "'" / "?" / "@" /
                       "^" / "`" / "~" / "*" / "$" / "\" /
                       "(" / ")" / "%" / "|" / "."

EQUAL                = LWSP %x3D LWSP ; "="
COLON                = %x3A           ; ":"
LBRKT                = LWSP %x7B LWSP ; "{"
RBRKT                = LWSP %x7D LWSP ; "}"
COMMA                = LWSP %x2C LWSP ; ","
DOT                  = %x2E           ; "."
SLASH                = %x2F           ; "/"
ALPHA                = %x41-5A / %x61-7A ; A-Z / a-z
DIGIT                = %x30-39         ; 0-9
DQUOTE               = %x22            ; " (Double Quote)
HEXDIG               = ( DIGIT / "A" / "B" / "C" / "D" / "E" / "F" )
SP                   = %x20        ; space
HTAB                 = %x09        ; horizontal tab
CR                   = %x0D        ; Carriage return
LF                   = %x0A        ; linefeed
LWSP                 = *( WSP / COMMENT / EOL )
EOL                  = (CR [LF] / LF )
WSP                  = SP / HTAB ; white space
SEP                  = ( WSP / EOL / COMMENT) LWSP
COMMENT              = ";" *(SafeChar/ RestChar / WSP / %x22) EOL
RestChar             = ";" / "[" / "]" / "{" / "}" / ":" / "," / "#" /
                       "<" / ">" / "="

; New Tokens added to sigParameter must take the format of SPA*
; * may be of any form i.e. SPAM
; New Tokens added to eventParameter must take the form of EPA*
; * may be of any form i.e. EPAD

AddToken                  = ("Add"                  / "A")
AuditToken                = ("Audit"                / "AT")
AuditCapToken             = ("AuditCapability"      / "AC")
AuditValueToken           = ("AuditValue"           / "AV")
AuthToken                 = ("Authentication"       / "AU")
BothwayToken              = ("Bothway"              / "BW")
BriefToken                = ("Brief"                / "BR")
BufferToken               = ("Buffer"               / "BF")
CtxToken                  = ("Context"              / "C")
ContextAuditToken         = ("ContextAudit"         / "CA")
DigitMapToken             = ("DigitMap"             / "DM")
DisconnectedToken         = ("Disconnected"         / "DC")
DelayToken                = ("Delay"                / "DL")
DurationToken             = ("Duration"             / "DR")
```

```
EmbedToken                  = ("Embed"                  / "EM")
EmergencyToken              = ("Emergency"              / "EG")
EmergencyOffToken           = ("EmergencyOffToken"      / "EGO")
ErrorToken                  = ("Error"                  / "ER")
EventBufferToken            = ("EventBuffer"            / "EB")
EventsToken                 = ("Events"                 / "E")
FailoverToken               = ("Failover"               / "FL")
ForcedToken                 = ("Forced"                 / "FO")
GracefulToken               = ("Graceful"               / "GR")
H221Token                   = ("H221" )
H223Token                   = ("H223" )
H226Token                   = ("H226" )
HandOffToken                = ("HandOff"                / "HO")
ImmAckRequiredToken         = ("ImmAckRequired"         / "IA")
InactiveToken               = ("Inactive"               / "IN")
IsolateToken                = ("Isolate"                / "IS")
InSvcToken                  = ("InService"              / "IV")
InterruptByEventToken       = ("IntByEvent"             / "IBE")
InterruptByNewSignalsDescrToken
                            = ("IntBySigDescr"          / "IBS")
KeepActiveToken             = ("KeepActive"             / "KA")
LocalToken                  = ("Local"                  / "L")
LocalControlToken           = ("LocalControl"           / "O")
LockStepToken                  = ("LockStep"            / "SP")
LoopbackToken               = ("Loopback"               / "LB")
MediaToken                  = ("Media"                  / "M")
MegacopToken                = ("MEGACO"                 / "!")
MethodToken                 = ("Method"                 / "MT")
MgcIdToken                  = ("MgcIdToTry"             / "MG")
ModeToken                   = ("Mode"                   / "MO")
ModifyToken                 = ("Modify"                 / "MF")
ModemToken                  = ("Modem"                  / "MD")
MoveToken                   = ("Move"                   / "MV")
MTPToken                    = ("MTP")
MuxToken                    = ("Mux"                    / "MX")
NotifyToken                 = ("Notify"                 / "N")
NotifyCompletionToken       = ("NotifyCompletion"/ "NC")
Nx64kToken                  = ("Nx64Kservice"           / "N64")
ObservedEventsToken         = ("ObservedEvents"         / "OE")
OnewayToken                 = ("Oneway"                 / "OW")
OnOffToken                  = ("OnOff"                  / "OO")
OtherReasonToken            = ("OtherReason"            / "OR")
OutOfSvcToken               = ("OutOfService"           / "OS")
PackagesToken               = ("Packages"               / "PG")
PendingToken                = ("Pending"                / "PN")
PriorityToken               = ("Priority"               / "PR")
ProfileToken                = ("Profile"                / "PF")
ReasonToken                 = ("Reason"                 / "RE")
RecvonlyToken               = ("ReceiveOnly"            / "RC")
ReplyToken                  = ("Reply"                  / "P")
RestartToken                = ("Restart"                / "RS")
RemoteToken                 = ("Remote"                 / "R")
ReservedGroupToken          = ("ReservedGroup"          / "RG")
ReservedValueToken          = ("ReservedValue"          / "RV")
SendonlyToken               = ("SendOnly"               / "SO")
SendrecvToken               = ("SendReceive"            / "SR")
ServicesToken               = ("Services"               / "SV")
ServiceStatesToken          = ("ServiceStates"          / "SI")
ServiceChangeToken          = ("ServiceChange"          / "SC")
ServiceChangeAddressToken   = ("ServiceChangeAddress"   / "AD")
SignalListToken             = ("SignalList"             / "SL")
SignalsToken                = ("Signals"                / "SG")
SignalTypeToken             = ("SignalType"             / "SY")
StatsToken                  = ("Statistics"             / "SA")
```

```
StreamToken                = ("Stream"                / "ST")
SubtractToken              = ("Subtract"              / "S")
SynchISDNToken             = ("SynchISDN"             / "SN")
TerminationStateToken      = ("TerminationState"      / "TS")
TestToken                  = ("Test"                  / "TE")
TimeOutToken               = ("TimeOut"               / "TO")
TopologyToken              = ("Topology"              / "TP")
TransToken                 = ("Transaction"           / "T")
ResponseAckToken               = ("TransactionResponseAck"/ "K")
V18Token                   = ("V18")
V22Token                   = ("V22")
V22bisToken                = ("V22b")
V32Token                   = ("V32")
V32bisToken                = ("V32b")
V34Token                   = ("V34")
V76Token                   = ("V76")
V90Token                   = ("V90")
V91Token                   = ("V91")
VersionToken               = ("Version"               / ——"V")
```

•••••

9) Anexo C Rótulos para las propiedades de los trenes de medios

•••••

Modifíquese el anexo C como sigue:

C.1 Atributos generales de los medios

Identificador de propiedad	Rótulo de propiedad	Tipo	Valor

•••••

Ganancia	100C	~~Entero sin signo~~	No se utiliza. Véase en E.13 una propiedad de ganancia disponible ~~Ganancia en dB: 0..65535~~

•••••

Ptime	1010	Entero	Tiempo de paquetización Establece la longitud de tiempo, en milisegundos, representada por los medios en un paquete Ref.: RFC 2327 del IETF

C.2 Propiedades de los múltiplex

•••••

C.12 H.245

Identificador de propiedad	Rótulo de propiedad	Tipo	Valor

•••••

CLCack	C006	Cadena de octetos	Valor de la estructura de H.245 CloseLogicalChannelAck Ref.: Rec. UIT-T H.245
LNC	C007	Entero	Valor del número de canal local de H.245, 0-65535. Ref.: Rec. UIT-T H.245

Anexo D

Transporte por IP

•••••

10) Anexo E Lotes básicos

•••••

Modifíquese el anexo E como sigue:

E.11 Lote de red

PackageID: nt (0x000b)

Versión: 1

Extiende: Ninguno

Este lote define propiedades de terminaciones de red independientes del tipo de red. Esto incluye, aunque no solamente, TDM, IP y ATM.

•••••

E.11.4 Estadísticas

Duración

StatisticsID: dur (0x0001)

Descripción: proporciona el lapso de tiempo durante el cual la terminación ha existido o ha estado fuera del contexto nulo ~~ha estado en el contexto~~.

Tipo: doble, en milisegundos

•••••

E.12.4 Estadísticas

•••••

Fluctuación

StatisticID: jit (0x0007)

Pide el valor actual de la fluctuación entre llegadas en un tren RTP como se define en RFC 1889. La fluctuación mide la variación del tiempo entre llegadas para paquetes de datos RTP.

Tipo: doble

Valores posibles: cualquier entero de 64 bits

Retardo

StatisticID: delay (0x0008)

Pide el valor actual del tiempo de propagación de paquete en unidades de indicación de tiempo. Equivale a la latencia promedio.

Tipo: doble

Valores posibles: cualquier entero de 64 bits

•••••

E.13.1 Propiedades

•••••

Control de ganancia

PropertyID: gain (0x000a)

El control de ganancia o la adaptación del nivel de señal y la reducción del nivel de ruido se utilizan para adaptar el nivel de la señal saliente. Sin embargo es necesario desactivar esta función, por ejemplo en llamadas en que intervienen módems. Cuando el valor se pone a "automático", la terminación ejerce una función de control de nivel automático (ALC), la MG proporciona el nivel objetivo, y la dirección es hacia afuera.

Tipo: entero

Valores posibles:

El control de ganancia especifica la ganancia en decibelios (positiva o negativa) reservándose el entero positivo máximo 214748647 (0x7fffffff) para representar "automático". ~~El parámetro control de ganancia puede especificarse, sea como "automático" (0xffffffff), sea como un número explícito de decibelios de ganancia (cualquier otro valor entero).~~ El valor por defecto es proporcionado por la MG.

Definido en: LocalControlDescriptor

Características: lectura/escritura.

•••••

11) Apéndice I Ejemplo de flujos de llamadas

•••••

Modifíquese el apéndice I como sigue:

I.1.1 Programación de terminaciones de línea analógica de pasarela residencial para comportamiento en reposo

A continuación se presentan las invocaciones de API por el controlador de pasarela de medios y por las pasarelas de medios para programar las terminaciones de este ejemplo para comportamiento en reposo. Tanto la pasarela de medios de origen como la de terminación tienen terminaciones AnalogLine en reposo programadas para que busquen eventos de iniciación de llamada (es decir, eventos de descolgado), utilizando la instrucción modificar con los parámetros apropiados. Se utiliza el contexto nulo para indicar que las terminaciones todavía no participan en un contexto. Se utiliza la terminación ROOT para indicar que se trata de la MG completa y no de una terminación dentro de la MG.

En este ejemplo, MG1 tiene la dirección IP 124.124.124.222, MG2 tiene la dirección IP 125.125.125.111, y el MGC tiene la dirección IP 123.123.123.4. El puerto Megaco por defecto es 55555 para los tres.

1) Una MG se registra en un MGC utilizando la instrucción ServiceChange:

```
MG1 to MGC:

MEGACO/1 [124.124.124.222]
Transaction = 9998 {
    Context = - {
        ServiceChange = ROOT {Services {
            Method=Restart, Version=2,
            ServiceChangeAddress=55555, Profile=ResGW/1}
        }
    }
}
```

2) El MGC envía una contestación:

```
MGC to MG1:
MEGACO/1 [123.123.123.4]:55555
Reply = 9998 {
   Context = - {ServiceChange = ROOT {
     Services {ServiceChangeAddress=55555, Profile=ResGW/1} } }
}
```

3) El MGC programa una terminación en el contexto NULL. El terminationId es A4444; el streamId es 1; el requestId en el descriptor de eventos es 2222. El mId es el identificador del emisor de este mensaje; en este caso, es la dirección IP y puerto [123.123.123.4]:55555. El modo para este tren se fija a SendReceive. "al" es el lote de supervisión de línea analógica. Se supone que se han configurado Local y Remote.

```
MGC to MG1:
MEGACO/~~1~~2 [123.123.123.4]:55555
Transaction = 9999 {
    Context = - {
        Modify = A4444 {
            Media { Stream = 1 {
                      LocalControl {
                          Mode = SendReceive,
                          tdmc/gain=2,  ; in dB,
                          tdmc/ec=on
                      },
                  }
            },
            Events = 2222 {al/of ~~+~~{strict=state}~~+~~}
        }
    }
}
```

El guión del plan de marcación podría haberse cargado previamente en la MG. Su función sería estar en espera de descolgado, activar el tono de invitación a marcar y comenzar a tomar los dígitos DTMF. Sin embargo, en este ejemplo se ha utilizado el mapa de dígitos, que se coloca en posición después al detectarse el evento de descolgado [paso 5) más adelante].

Obsérvese que el EventsDescriptor insertado podría haberse utilizado para combinar los pasos 3) y 4) con los pasos 8) y 9), y eliminar así los pasos 6) y 7).

4) El MG1 acepta la instrucción Modificar enviando esta contestación:

```
MG1 to MGC:
MEGACO/~~1~~2 [124.124.124.222]:55555
Reply = 9999 {
   Context = - {Modify = A4444}
}
```

5) Un intercambio similar tiene lugar entre la MG2 y el MGC, obteniéndose como resultado una terminación en reposo denominada A5555.

I.1.2 Toma de dígitos del originador e iniciación de la terminación

Lo siguiente se basa en las condiciones anteriormente mostradas. Se presentan las transacciones procedentes del controlador de pasarela de medios y de la pasarela de medios de origen (MG1, *originating media gateway*) para hacer que la terminación de origen (A4444) pase a través de las etapas de toma de dígitos requeridas para iniciar una conexión con la pasarela de medios de terminación (MG2, *terminating media gateway*).

6) La MG1 detecta un evento de descolgado procedente del usuario 1 y lo informa al controlador de pasarela de medios mediante una instrucción Notificar.

```
MG1 to MGC:
MEGACO/1 2 [124.124.124.222]:55555
Transaction = 10000 {
   Context = - {
       Notify = A4444 {ObservedEvents =2222 {
         19990729T22000000:al/of(init=false)}}
   }
}
```

7) Y se acusa recibo de la instrucción Notificar.

```
MGC to MG1:
MEGACO/1 2 [123.123.123.4]:55555
Reply = 10000 {
    Context = - {Notify = A4444}
}
```

8) El MGC modifica la terminación para reproducir el tono de marcar, buscar dígitos de conformidad con Dialplan0 y buscar el evento colgado ahora.

```
MGC to MG1:
MEGACO/1 2 [123.123.123.4]:55555
Transaction = 10001 {
    Context = - {
        Modify = A4444 {
            Events = 2223 {
                al/on(strict=state), dd/ce {DigitMap=Dialplan0}
            },
            Signals {cg/dt},
            DigitMap= Dialplan0{
(0| 00|[1-7]xxx|8xxxxxxx|Fxxxxxxx|Exx|91xxxxxxxxxx|9011x.)}
        }
    }
}
```

9) Y se acusa recibo de la instrucción Modificar.

```
MG1 to MGC:
MEGACO/1 2 [124.124.124.222]:55555
Reply = 10001 {
    Context = - {Modify = A4444}
}
```

10) Seguidamente, la MG1 acumula los dígitos a medida que son marcados por el usuario 1. Cuando se obtiene una concordancia apropiada de los dígitos tomados con el plan de marcación de números para A4444, se envía una nueva instrucción Notificar al controlador de pasarela de medios.

```
MG1 to MGC:
MEGACO/1 2 [124.124.124.222]:55555
Transaction = 10002 {
   Context = - {
       Notify = A4444 {ObservedEvents =2223 {
         19990729T22010001:dd/ce{ds="916135551212",Meth=UM}}}
   }
}
```

11) Y se acusa recibo de la instrucción Notificar.

```
MGC to MG1:
MEGACO/1 2 [123.123.123.4]:55555
Reply = 10002 {
```

```
    Context = - {Notify = A4444}
}
```

12) El controlador analiza entonces los dígitos y determina que es necesario establecer una conexión de MG1 a MG2. Tanto la terminación A4444 como la terminación RTP se añaden a un nuevo contexto en MG1. Todavía está en modo ReceiveOnly porque no se han especificado valores de descriptor distante. Los códecs preferidos están en el orden de preferencia por el MGC.

```
MGC to MG1:
MEGACO/~~1~~2 [123.123.123.4]:55555
Transaction = 10003 {
    Context = $ {
       Add = A4444,
       Add = $ {
           Media {
             Stream = 1 {
                  LocalControl {
                      Mode = ReceiveOnly,

                      nt/jit=40 ; in ms
                  },
                  Local {
v=0
c=IN IP4 $
m=audio $ RTP/AVP 4
a=ptime:30
v=0
c=IN IP4 $
m=audio $ RTP/AVP 0
                  }
             }
          }
       }
    }
}
```

NOTA – El MGC expresa sus valores de parámetros preferidos como una serie de bloques SDP en Local. La MG llena el descriptor Local en la contestación.

13) La MG1 acusa recibo de la nueva terminación y llena la dirección IP local en el puerto UDP. También elige un códec basándose en las preferencias del MGC en local. La MG1 fija el puerto RTP a 2222.

```
MEGACO/~~1~~2 [124.124.124.222]:55555
Reply = 10003 {
   Context = 2000 {
      Add = A4444,
      Add=A4445{
         Media {
             Stream = 1 {
                 Local {
v=0
o=- 2890844526 2890842807 IN IP4 124.124.124.222
s=-
t= 0 0
c=IN IP4 124.124.124.222
m=audio 2222 RTP/AVP 4
a=ptime:30
a=recvonly
```

```
                } ; RTP profile for G.723.1 is 4
             }
          }
       }
    }
}
```

14) El MGC asocia ahora A5555 con el nuevo contexto en MG2, y establece un tren RTP (esto es, se reasigna A5556), quedando establecida la conexión SendReceive hasta el usuario de origen, usuario 1. El MGC también arregla el timbre en A5555.

```
MGC to MG2:
MEGACO/~~1~~2 [123.123.123.4]:55555
Transaction = 50003 {
    Context = $ {
       Add = A5555  { Media {
            Stream = 1 {
                 LocalControl {Mode = SendReceive} }},
          Events=1234{al/of(strict=state)},
            Signals {al/ri}
            },
       Add  = $ {Media {
            Stream = 1 {
                 LocalControl {
                    Mode = SendReceive,
                    nt/jit=40 ; in ms
                 },
                 Local {
v=0
c=IN IP4 $
m=audio $ RTP/AVP 4
a=ptime:30
                 },
                 Remote {
v=0
c=IN IP4 124.124.124.222
m=audio 2222 RTP/AVP 4
a=ptime:30
                 } ; RTP profile for G.723.1 is 4
             }
          }
       }
    }
}
```

15) Se acusa recibo. El número de puerto de tren es diferente del número del puerto de control. En este caso es 1111 (en SDP).

```
MG2 to MGC:
MEGACO/~~1~~2 [125.125.125.111]:55555
Reply = 50003 {
   Context = 5000 {
     Add = A5555,
       Add = A5556{
          Media {
             Stream = 1 {
                 Local {
v=0

o=- 7736844526 7736842807 IN IP4 125.125.125.111
s=-
t= 0 0
c=IN IP4 125.125.125.111
```

```
m=audio 1111 RTP/AVP 4
}
            } ; RTP profile for G.723.1 is 4
          }
        }
    }
}
```

16) Ahora es necesario comunicar a MG1 la dirección IP y el puerto UDP antes mencionados.

```
MGC to MG1:
MEGACO/~~1~~2 [123.123.123.4]:55555
Transaction = 10005 {
  Context = 2000 {
    Modify = A4444 {
      Signals {cg/rt}
    },
    Modify = A4445 {
       Media {
            Stream = 1 {
                Remote {
v=0

o=- 7736844526 7736842807 IN IP4 125.125.125.111
s=-
t= 0 0
c=IN IP4 125.125.125.111
m=audio 1111 RTP/AVP 4
                }
            } ; RTP profile for G.723.1 is 4
        }
    }
  }
}

MG1 to MGC:
MEGACO/~~1~~2 [124.124.124.222]:55555
Reply = 10005 {
   Context = 2000 {Modify = A4444, Modify = A4445}
}
```

17) Las dos pasarelas están ahora conectadas y el usuario 1 oye el tono de llamada. La MG2 espera ahora hasta que el usuario 2 descuelgue, en cuyo momento quedará establecida la comunicación en ambos sentidos.

```
From MG2 to MGC:

MEGACO/~~1~~2 [125.125.125.111]:55555
Transaction = 50005 {
   Context = 5000 {
       Notify = A5555 {ObservedEvents =1234 {
         19990729T22020002:al/of(init=false)}}
   }
}

From MGC to MG2:

MEGACO/~~1~~2 [123.123.123.4]:55555
Reply = 50005 {
    Context = - {Notify = A5555}
}

From MGC to MG2:

MEGACO/~~1~~2 [123.123.123.4]:55555
```

```
Transaction = 50006 {
   Context = 5000 {
      Modify = A5555 {
         Events = 1235 {al/on(strict=state)},
          Signals { } ; to turn off ringing
      }
   }
}

From MG2 to MGC:

MEGACO/1 2 [125.125.125.111]:55555
Reply = 50006 {
 Context = 5000 {Modify = A4445}
}
```

18) Se cambia el modo en MG1 a SendReceive, y se detiene la señal de llamada de retorno.

```
MGC to MG1:
MEGACO/1 2 [123.123.123.4]:55555
Transaction = 10006 {
   Context = 2000 {
      Modify = A4445 {
         Media {
            Stream = 1 {
               LocalControl {
                  Mode=SendReceive
               }
            }
         }
      },
      Modify = A4444 {
         Signals { }
      }
   }
}

from MG1 to MGC:
MEGACO/1 2 [124.124.124.222]:55555
Reply = 10006 {
   Context = 2000 {Modify = A4445, Modify = A4444}}
```

19) El MGC decide efectuar una auditoría en la terminación RTP en MG2.

```
MEGACO/1 2 [123.123.123.4]:55555
Transaction = 50007 {
   Context = - {AuditValue = A5556{
      Audit{Media, DigitMap, Events, Signals, Packages, Statistics }}
   }
}
```

20) El MG2 contesta.

```
MEGACO/1 2 [125.125.125.111]:55555
Reply = 50007 {
   Context = - {
AuditValue = A5556 {
          Media {
             TerminationState { ServiceStates = InService,
                 Buffer = OFF },
             Stream = 1 {
                 LocalControl { Mode = SendReceive,
                    nt/jit=40 },
                 Local {
```

```
v=0

o=- 7736844526 7736842807 IN IP4 125.125.125.111
s=-
t= 0 0
c=IN IP4 125.125.125.111
m=audio 1111 RTP/AVP  4
a=ptime:30
                },
                 Remote {
v=0

o=- 2890844526 2890842807 IN IP4 124.124.124.222
s=-
t= 0 0
c=IN IP4 124.124.124.222
m=audio 2222 RTP/AVP  4
a=ptime:30
                 } } },
          Events,
          Signals,
          DigitMap,
          Packages {nt-1, rtp-1},
          Statistics { rtp/ps=1200,  ; packets sent
                       nt/os=62300, ; octets sent
                       rtp/pr=700, ; packets received
                       nt/or=45100, ; octets received
                       rtp/pl=0.2,  ; % packet loss
                       rtp/jit=20,
                       rtp/delay=40 } ; avg latency
       }
    }
}
```

21) Cuando el MGC recibe una señal de colgado de una de las MG, suprime la llamada. En este ejemplo, el primero que cuelga es el usuario en MG2.

```
From MG2 to MGC:

MEGACO/~~1~~2 [125.125.125.111]:55555
Transaction = 50008 {
   Context = 5000 {
       Notify = A5555 {ObservedEvents =1235 {
          19990729T24020002:al/on(init=false)}
       }
   }
}

From MGC to MG2:

MEGACO/~~1~~2 [123.123.123.4]:55555
Reply = 50008 {
    Context = - {Notify = A5555}
}
```

22) El MGC envía ahora a ambas MG una instrucción Substraer para suprimir la llamada. En este ejemplo sólo se muestra la instrucción Substraer enviada a MG2. Cada terminación reúne su propio conjunto de estadísticas. Es posible que el MGC no tenga que pedir que se le envíen ambos conjuntos. A5555 es una terminación física, y A5556 es una terminación RTP.

```
From MGC to MG2:

MEGACO/~~1~~2 [123.123.123.4]:55555
```

```
Transaction = 50009 {
   Context = 5000 {
      Subtract = A5555 {Audit{Statistics}},
      Subtract = A5556 {Audit{Statistics}}
   }
}

From MG2 to MGC:

MEGACO/1 2 [125.125.125.111]:55555
Reply = 50009 {
   Context = 5000 {
     Subtract = A5555 {
          Statistics {
             nt/os=45123, ; Octets Sent
             nt/or=45123, ; Octets Received
             nt/dur=40000 ; in milliseconds
             }
       },
       Subtract = A5556 {
          Statistics {
             rtp/ps=1245, ; packets sent
             nt/os=62345, ; octets sent
             rtp/pr=780, ; packets received
             nt/or=45123, ; octets received
             rtp/pl=10, ; -% packets lost
             rtp/jit=27,
             rtp/delay=48, ; average latency
             nt/dur=38000  ; in millisec
          }
       }
   }
}
```

23) El MGC configura ahora ambas pasarelas, MG1 y MG2, de modo que estén listas para la detección del próximo evento de descolgado. Véase el paso 1). Obsérvese que éste podría ser el estado por defecto de una terminación en el contexto nulo, en cuyo caso el MGC no tendría que enviar ningún mensaje a la MG. La terminación que retorna al contexto nulo toma de nuevo sus valores por defecto.

SERIES DE RECOMENDACIONES DEL UIT-T

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

Impreso en Suiza
Ginebra, 2004