Unión Internacional de Telecomunicaciones

UIT-T

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

J.203

(11/2006)

SERIE J: REDES DE CABLE Y TRANSMISIÓN DE PROGRAMAS RADIOFÓNICOS Y TELEVISIVOS, Y DE OTRAS SEÑALES MULTIMEDIA

Aplicación para televisión digital interactiva

Núcleo común para plataformas de grabación de vídeo digital

Recomendación UIT-T J.203

Unión Internacional de Telecomunicaciones

Recomendación UIT-T J.203

Núcleo común para plataformas de grabación de vídeo digital

Resumen

En esta Recomendación se definen las interfaces de programas de aplicación (API, *application programme interface*), garantías semánticas y otros aspectos del sistema asociado a plataformas de grabación de vídeo digital armonizada.

Orígenes

La Recomendación UIT-T J.203 fue aprobada el 29 de noviembre de 2006 por la Comisión de Estudio 9 (2005-2008) 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 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 en la dirección http://www.itu.int/ITU-T/ipr/.

ÍNDICE

Página

Recomendación UIT-T J.203

Núcleo común para plataformas de grabación de vídeo digital

1 Ámbito

En esta Recomendación se describe una ampliación modular a [UIT-T J.202], actualizada de conformidad con GEM [ETSI TS 102 819], que describe como debe integrarse la grabación y reproducción de contenidos de vídeo (y audio) digital con la plataforma GEM [ETSI TS 102 819]. Esta Recomendación está destinada, en primer lugar, a entidades que desarrollan especificaciones de terminales y/o normas que amplían la especificación de un terminal GEM [ETSI TS 102 819] con grabación y reproducción de vídeo (y audio) digital. En segundo lugar, a desarrolladores de aplicaciones GEM [ETSI TS 102 819] que deseen utilizar la grabación y reproducción de vídeo (y audio) digital. Los implementadores deberían consultar con quienes publican especificaciones que hacen referencia a GEM [ETSI TS 102 819] en relación con la conformidad de la misma.

NOTA – En esta Recomendación se definen las interfaces visibles a las aplicaciones. Los desarrolladores de aplicaciones no deberían asumir que cualquier interfaz conexa está disponible, salvo que se mencione explícitamente. La normativa relativa a terminales o las implementaciones de éstos pueden incluir otras interfaces. Uno de los objetivos básicos de esta Recomendación es maximizar los aspectos comunes relativos a la integración de grabación digital de audio y vídeo entre MHP [ETSI ES 201 812] y las diversas especificaciones de terminales GEM [ETSI TS 102 819].

2 Referencias

2.1 Referencias normativas

Las siguientes Recomendaciones del UIT-T y otras referencias contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y otras referencias son objeto de revisiones por lo que se preconiza que los usuarios de esta Recomendación investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y otras referencias citadas a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T actualmente vigentes. En esta Recomendación, la referencia a un documento, en tanto que autónomo, no le otorga el rango de una Recomendación.

[ITU-T J.202] Recomendación UIT-T J.202 (2005), *Armonización de los formatos de contenidos de procedimiento para las aplicaciones de televisión interactiva.*

[ETSI ES 201 812] ETSI ES 201 812 V1.1.1, Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0.3.

[ETSI TS 102 812] ETSI TS 102 812 V1.3.1, Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.2.

[ETSI TS 102 819] ETSI TS 102 819, Digital Video Broadcasting (DVB); Globally Executable MHP (GEM).

2.2 Referencias informativas

Aunque las referencias siguientes tienen carácter informativo a fin de no restringir la aplicación de esta Recomendación a servicios del tipo "TV-Anytime", se insta a los usuarios que utilicen las normas siguientes en aras de la armonización.

– ETSI TS 102 822-1 V1.3.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 1: Benchmark Features.

– ETSI TS 102 822-2 V1.3.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 2: System description.

– ETSI TS 102 822-3-1 V1.3.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 1: Phase 1 – Metadata schemas.

– ETSI TS 102 822-3-2 V1.3.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 2: System aspects in a uni-directional environment.

– ETSI TS 102 822-3-3 V1.1.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 3: Phase 2 – Extended Metadata Schema.

– ETSI TS 102 822-3-4 V1.1.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 4: Phase 2 – Interstitial metadata.

– ETSI TS 102 822-4 V1.2.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 4: Content referencing.

– ETSI TS 102 822-5 V1.1.1 (2005-03), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime Phase 1"); Part 5: Rights Management and Protection (RMP) Information for Broadcast Applications.

– ETSI TS 102 822-5-1 V1.2.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 5: Rights Management and Protection (RMP) Sub-part 1: Information for Broadcast Applications.

– ETSI TS 102 822-5-2 V1.2.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 5: Rights Management and Protection (RMP) Sub-part 2: RMPI binding.

– ETSI TS 102 822-6-1 V1.3.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 6: Delivery of metadata over a bi-directional network; Sub-part 1: Service and transport.

– ETSI TS 102 822-6-2 V1.3.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 6: Delivery of metadata over a bi-directional network; Sub-part 2: Phase 1 – Service discovery.

– ETSI TS 102 822-6-3 V1.1.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 6: Delivery of metadata over a bi-directional network; Sub-part 3: Phase 2 – Exchange of Personal Profile.

– ETSI TS 102 822-7 V1.1.1 (2003-10), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime Phase 1"); Part 7: Bi-directional metadata delivery protection.

– ETSI TS 102 822-8 V1.1.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 8: Phase 2 – Interchange Data Format.

– ETSI TS 102 822-9 V1.1.1 (2006-01), Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 9: Phase 2 – Remote Programming.

3 Definiciones

En esta Recomendación se definen los términos siguientes:

3.1 especificación de grabación GEM: Especificación completa que amplía este documento con el objetivo de crear una especificación completa sobre cómo integrar la grabación y reproducción de vídeo (y audio) digital con las especificaciones de terminal GEM.

3.2 terminal de grabación GEM: Terminal u otro dispositivo que es conforme con una especificación de grabación GEM.

3.3 reproducción normal: Reproducción de contenidos de vídeo y/o audio en la dirección hacia adelante y con una velocidad de 1,0. Incluye contenidos en directo, contenido en directo con desplazamiento temporal y contenido consecuencia de una grabación programada.

3.4 flujos grabables: Flujo de datos que forma parte de un contenido que debe grabarse y reproducirse. En el caso de especificaciones de terminal GEM, incluye flujos que se identifica que se corresponden con lo indicado en las cláusulas 7.2.1 ("Audio") y 7.2.2 ("Video") de GEM. Las especificaciones de grabación GEM [ETSI TS 102 819] pueden definir que otros tipos de flujos sean flujos grabables, por ejemplo, los subtítulos o los subtítulos ocultos para sordos.

3.5 grabación: Un término genérico que hace referencia a un concepto de gran público para un evento programado o grabado y que no se refiere a un objeto Java real. El término abarca grabaciones solicitadas pero aún no comenzadas, grabaciones en curso, grabaciones completadas (con éxito o no) y grabaciones infructuosas, de las que no se ha grabado ningún dato.

3.6 grabaciones programadas: Grabaciones para las que el objeto de la grabación se especifica mediante la hora, duración y canal, o mediante un identificador que se traduce en tiempo, duración y canal.

3.7 secuencia temporal sintetizada: Secuencia temporal de un contenido específico sintetizado mediante el terminal de grabación GEM (en contraposición a ser incluido en el contenido cuando ha sido transmitido).

3.8 secuencia temporal: Paso del tiempo inherentemente asociado a un elemento de contenido al que pueden hacer referencia aplicaciones y que puede distribuirse mediante una secuencia temporal transmitida.

3.9 grabaciones con desplazamiento de tiempo: Grabaciones cuyo objetivo es un contenido de audio y vídeo que ya se ha recibido.

3.10 secuencia temporal transmitida: Una secuencia temporal que forma parte de un contenido cuando éste se transmite (por ejemplo, tiempo de reproducción normal de DSMCC).

3.11 reproducción en modo o modos truco: Término genérico que se utiliza para hacer referencia a la reproducción de vídeo y/o audio a velocidades distintas a la normal (1,0), o en el sentido contrario al avance de la reproducción. Incluye la reproducción a velocidad baja y alta, hacia adelante y hacia atrás, así como la pausa.

3.12 aplicación consciente del modo de reproducción truco: Aplicación que se identifica conocedora de utilizar el modo de reproducción truco, es decir, con la bandera trick_mode_aware_flag puesta a '1'.

4 Abreviaturas

En esta Recomendación se utilizan las abreviaturas siguientes:

AIT Tabla de información de aplicación (*application information table*), tal como se define en la cláusula 11.4 de [ETSI ES 201 812] y [ETSI TS 102 812].

DVR Grabador de vídeo digital (*digital video recorder*)

PDR Grabador personal digital (*personal digital recorder*)

5 Convenios

El término "aplicación" se utiliza para aplicaciones GEM y para aplicaciones que no lo son en sí mismas pero que son conformes con la especificación de terminal GEM implementada mediante una determinada terminal de grabación GEM.

En esta Recomendación se utiliza la terminología de que algo "no deberá ser grabado" o "no debería ser grabado". Se trata de un convenio de términos en aras de la brevedad y en todos los casos, se permite que las implementaciones graben elementos que "no deberán" o "no deberían ser grabados" y se descartan durante la reproducción. El término "no deberá ser grabado" es sencillamente más breve que "no deberá ser grabado o si se graba, deberá descartarse durante la reproducción", siendo esto último lo que realmente se quiere decir.

6 Consideraciones generales

6.1 Objetivo

Esta Recomendación no pretende ser, y no debería utilizarse como, una especificación completa de cómo integrar vídeo (y audio) digital y reproducirlo utilizando especificaciones de terminal GEM. Es un marco sobre el que puede crearse una especificación completa (conocida como especificación de grabación GEM).

6.2 Conformidad total con esta Recomendación

Cuadro 6-1 – Especificaciones de terminales MHP y GEM

[ETSI ES 201 812]	MHP 1.0
[ETSI TS 102 812]	MHP 1.1
[ETSI TS 102 819]	GEM

Cuadro 6-2 – Especificaciones de grabación MHP y GEM

[b-ETSI TS 102 816]	PVR/PDR Extension to the Multimedia Home Platform
[b-OC-SP-OCAP-DVR-I02-050524]	OCAP Digital Video Recorder

A fin evitar ambigüedades, cualquier equipo que sea conforme con la totalidad de esta Recomendación salvo con la cláusula anterior no se considera conforme con esta Recomendación.

7 Procesos de grabación y reproducción

Las implementaciones basadas en esta Recomendación realizarán las actuaciones siguientes como parte de sus procesos de grabación y reproducción.

7.1 Gestión de grabaciones programadas

El proceso de gestión de grabaciones programadas incluirá las actividades siguientes:

1) mantenimiento de una lista de peticiones de grabación que se mantienen en estado pendiente al menos hasta el final del periodo de validez de la petición de grabación;

2) mantenimiento de una lista de peticiones de grabación finalizadas con éxito, grabaciones que se iniciaron pero que no finalizaron con éxito y grabaciones que habían sido programadas pero que ni siquiera se iniciaron;

3) inicio del proceso de grabación de las peticiones de grabación pendientes en el momento adecuado, incluyendo, si fuera necesario, la activación desde un estado de reposo o similar en relación con la gestión de la alimentación de energía;

4) mantenimiento de referencias a las peticiones de la lista de peticiones de grabación que hayan fracasado, al menos durante el periodo de validez de la petición de grabación.

NOTA – Los mecanismos para resolver los conflictos entre las peticiones de grabación (por ejemplo, utilización del sintonizador) quedan fuera del ámbito de esta Recomendación y deberían incluirse en las especificaciones de grabación GEM.

7.2 Proceso de grabación

El proceso de grabación incluirá las actividades siguientes:

1) Identificar qué flujos grabables deberían grabarse. Si se especificaron flujos específicos cuando originalmente se programó la petición de grabación, esos son los que deberían grabarse. Si no se especificaron flujos concretos cuando originalmente se programó la petición de grabación, deberían grabarse los flujos por defecto del contenido en cuestión.

   NOTA 1 – La definición de flujos por defecto que han de ser grabados queda fuera del ámbito de esta Recomendación y deberían incluirse en las especificaciones de grabación GEM.

2) Grabar los flujos identificados hasta el límite de capacidad de grabación del terminal de grabación GEM. Cuando fuera necesario grabar un número de flujos de un tipo dado superior a lo que pueda el terminal de grabación GEM, los flujos se ordenarán con prioridad de acuerdo con la cláusula 11.4 de [ETSI ET 102 819] (que a su vez es conforme con la cláusula 11.4.2.3 de [ETSI ES 201 812] y de [ETSI TS 102 812]).

   NOTA 2 – Las capacidades mínimas de número de flujos de cada tipo que deben ser capaces de grabar los terminales de grabación GEM queda fuera del ámbito de esta Recomendación y deberían incluirse en las especificaciones de grabación GEM.

3) Identificar las aplicaciones grabables y grabarlas junto con datos suficientes para reconstruir sus anotaciones AIT.

   Las aplicaciones con una descripción de grabación de aplicación cuya bandera scheduled_recording_flag (bandera de grabación programada) esté puesta a 0 no se considerarán grabables.

   NOTA 3 – Una definición más completa sobre cuáles son las aplicaciones grabables (y cuáles no) queda fuera del ámbito de esta Recomendación y debería incluirse en las especificaciones de grabación GEM.

   NOTA 4 – Los requisitos de un terminal de grabación GEM para supervisar datos y comportamiento dinámico de aplicaciones durante una grabación quedan fuera del ámbito de esta Recomendación y deberían incluirse en las especificaciones de grabación GEM.

4) Grabar información suficiente sobre todas las secuencias temporales transmitidas que formen parte de las grabaciones a fin de reconstruirlas con precisión durante la reproducción.

5) Generar un tiempo de medios que se incremente de modo lineal a una velocidad de 1,0 desde el inicio hasta al fin de la grabación.

6) Manejar los casos siguientes relativos a la interrupción de la alimentación de energía a un terminal de grabación GEM durante una grabación programada:

   – la grabación está en curso y se suspende la alimentación de energía;

   – la grabación se ha iniciado cuando la alimentación de energía no estaba disponible pero ésta se ha reactivado antes de finalizar la grabación programada;

   – estando la grabación en curso, la alimentación de energía se suspende y vuelve una o más veces sucesivamente;

– la grabación se ha iniciado cuando la alimentación de energía no estaba disponible pero ésta vuelve y se suspende una o más veces sucesivamente.

En todos los casos, se grabará el contenido que esté disponible durante los periodos en que el terminal de grabación GEM esté alimentado.

7.3 Gestión de las grabaciones finalizadas

El proceso de gestión de grabaciones finalizadas incluirá las actividades siguientes:

1) Mantener junto con todas las grabaciones realizadas la información siguiente, en tanto que se mantenga el contenido;
   - si se tiene conocimiento de que la grabación ha sido finalizada o bien que haya quedado sin finalizar o si su estado es desconocido;
   - la hora y el canal sobre el que se hizo la grabación;
   - los datos específicos de la aplicación asociados a la grabación.
2) Suprimir, eventualmente, las grabaciones (incluido el apunte en la lista de grabaciones, los datos grabados y cualquier otra información asociada) una vez pasado el periodo de expiración para la petición de grabación hoja correspondiente a esta grabación.

   NOTA – Esta Recomendación no hace referencia alguna a la precisión con la que debe respetarse el periodo de expiración. Las especificaciones de grabación GEM deben especificar la precisión que debe ser exigida en las implementaciones.

7.4 Reproducción de grabaciones programadas

7.4.1 Proceso de reproducción

El proceso reproducción de grabaciones programadas incluirá las actividades siguientes:

1) Iniciar la reproducción de los flujos grabables salvo que la bandera initiating_replay_flag (bandera de inicio de reproducción) de la descripción de la grabación de aplicación esté puesta a "1".
2) Iniciar la reproducción de las aplicaciones grabadas cuando éstas formen parte del contenido grabado y su application_control_code (código de control de aplicación) sea AUTOSTART.

   NOTA – Los requisitos para reconstruir durante la reproducción el comportamiento dinámico de las aplicaciones grabadas quedan fuera del ámbito de esta Recomendación y deberían incluirse en las especificaciones de grabación GEM.
3) Cuando se está reproduciendo un contenido que está siendo grabado y finaliza dicho contenido deteniéndose la grabación, la reproducción debe continuar sin interrupción (aunque no necesariamente de una forma perfectamente imperceptible), con independencia de cualquier proceso (que sea función de la implementación concreta) destinado a copiar el contenido recién grabado desde una memoria intermedia temporal a una ubicación más permanente del dispositivo de almacenamiento.
4) Generar una referencia de tiempo que se incremente linealmente desde el inicio de la grabación hasta su final. Ésta se utilizará como la base del "tiempo de base de tiempo" y para el "tiempo de medios", tal como se define en el marco de medios Java. No es necesario que exista relación alguna entre este tiempo y cualquier otro tiempo que forme parte del contenido grabado, tal como MPEG PCR, STC o DSMCC NPT.
5) Para todas las secuencias temporales transmitidas que formen parte de la grabación original, se reconstruye cada secuencia temporal cuando el tiempo de medios esté comprendido en el rango de valores para el que dicha secuencia temporal es válida.

7.4.2 Eventos durante la reproducción

Durante la reproducción de un contenido grabado como consecuencia de una grabación programada, se soportará el comportamiento siguiente:

Cuadro 7-1 – Eventos durante la reproducción normal y comportamiento resultante

Evento	Comportamiento	Resultado en la pantalla
Avance rápido hasta el final del flujo	Fin del evento de medios generado para cualquiera de las aplicaciones registradas	Última trama congelada
Retroceso rápido hasta el inicio del flujo	Conmutación al modo pausa	Primera trama congelada
Reproducción hasta el final del flujo	Fin de los medios generados para cualquiera de las aplicaciones registradas	Última trama congelada

7.5 Desplazamiento temporal

7.5.1 Proceso de grabación

El proceso de grabación con desplazamiento temporal incluirá las actividades siguientes:

1) Identificar los flujos grabables que deben ser grabados y grabarlos.

   NOTA 1 – La definición de flujos que deben ser grabados con desplazamiento temporal queda fuera del ámbito de esta Recomendación y debería incluirse en las especificaciones de grabación GEM.

2) Identificar las aplicaciones grabables señalizadas cuyo contenido se graba al inicio de la grabación y grabar al menos las aplicaciones y datos suficientes para reconstruir sus anotaciones AIT.

   NOTA 2 – La definición de qué aplicaciones son grabables con desplazamiento temporal (y cuáles no) queda fuera del ámbito de esta Recomendación y deberían incluirse en las especificaciones de grabación GEM.

   NOTA 3 – Los requisitos de un terminal de grabación GEM que permitan supervisar datos y comportamiento dinámico de aplicaciones durante una grabación con desplazamiento temporal quedan fuera del ámbito de esta Recomendación y deberían incluirse en las especificaciones de grabación GEM.

3) Identificar las secuencias temporales que formen parte del contenido que se está grabando y grabar información suficiente sobre las mismas para permitir su reconstrucción precisa durante la reproducción.

4) Gestionar la memoria intermedia de desplazamiento temporal para que cuando ésta se llene, la grabación continúe en la memoria intermedia de forma que se sobrescriba sobre los contenidos más antiguos.

7.5.2 Reproducción

La reproducción de contenidos grabados en modo desplazamiento temporal requiere que una memoria intermedia esté asociada con un reproductor o contexto de servicio JMF. La definición de cómo y cuándo se hace esta asociación queda fuera del ámbito de esta Recomendación y debería incluirse en las especificaciones de grabación GEM

Durante la reproducción de contenidos grabados en modo desplazamiento temporal, el comportamiento será el que se define en la cláusula 7.4 con las modificaciones siguientes:

1) El comportamiento definido en el cuadro 7-1, Eventos durante la reproducción normal y comportamiento resultante, se sustituirá por el cuadro siguiente.

Cuadro 7-2 – Eventos durante la reproducción con desplazamiento temporal y comportamiento resultante

Evento	Comportamiento
Avance rápido hasta el fin de la memoria intermedia de desplazamiento temporal (es decir, el punto en el que se escribe el contenido recién grabado)	Conmuta a reproducción con velocidad = 1,0 en la grabación almacenada o en directo sin destruir el contenido de la memoria intermedia de desplazamiento temporal.
Retroceso rápido hasta el comienzo de la memoria intermedia de desplazamiento temporal (es decir, el punto en el que se sobrescribe contenido previamente grabado)	Conmuta a reproducción normal con velocidad = 1,0 en la grabación almacenada en la memoria intermedia, desde el inicio del contenido en la memoria intermedia de desplazamiento temporal.
Reproducir hasta el fin de la memoria intermedia de desplazamiento temporal (es decir, una mala recepción de señal produce la infrautilización de la capacidad de la memoria intermedia de desplazamiento temporal)	No se hace nada (es decir, continua la reproducción con velocidad = 1,0 en la grabación almacenada en la memoria intermedia) o bien, se conmuta a en directo, sin destruir el contenido de la memoria intermedia de desplazamiento temporal.
Pausa hasta que la memoria intermedia de desplazamiento temporal está "llena" y el contenido en el punto en el que se hizo la pausa va a comenzar a ser sobrescrito.	Conmuta a reproducción con velocidad = 1,0 desde el instante en que se hizo la pausa.

Para todas las secuencias temporales transmitidas que son válidas en la memoria intermedia de desplazamiento temporal, se reconstruye cada secuencia temporal cuando el tiempo de medios actual está dentro del margen para el que dicha secuencia temporal es válida.

2) El tiempo de medios para cada ubicación en la memoria intermedia de desplazamiento temporal será el valor asignado a dicha ubicación en el punto en que fue grabado. No habrá discontinuidades en el tiempo de medios en la grabación almacenada en la memoria temporal incluyendo la "cabecera" de la memoria temporal en que el contenido coincide con el contenido recibido en ese momento.

3) Cuando se fija el tiempo de medios a un valor correspondiente a POSITIVE_INFINITY, se fijará la ubicación de la reproducción en el punto de registro actual si la grabación aún está en curso, o, si no es así, al final de la grabación.

4) Si la posición de la reproducción coincide con el punto de grabación (es decir, se reproduce el punto del contenido en directo), el contenido se mostrará con un retardo cero en relación con el contenido difundido original. Las implementaciones que pueden reproducir el contenido grabado con un retardo nulo respecto a la reproducción en directo pueden reproducir el contenido grabado. Otras implementaciones deben reproducir el contenido difundido original.

8 API de grabación y de reproducción

8.1 Grabación y gestión de la grabación

8.1.1 Visión general (informativa)

Las API de grabación permiten que las aplicaciones realicen la grabación de dos formas posibles:

1) Programar una grabación que se realizará en un momento posterior.

2) Grabar eventos difundidos en tiempo real. Ello también puede incluir la parte del evento de difusión que está en la memoria intermedia de desplazamiento temporal.

La implementación mantiene una base de datos de grabaciones, representada por el singleton RecordingManager (un singleton es un tipo de objeto utilizado en técnicas de programación del que sólo se puede crear una instancia). Las aplicaciones crean grabaciones llamando al método "record()" del objeto RecordingManager (gestor de grabación). Esta operación crea una anotación en la base de datos de la implementación. Dichas anotaciones se representan mediante instancias de RecordingRequest (petición de grabación). Las aplicaciones pueden asociar datos específicos de la aplicación con RecordingRequests utilizando para ello el método "addAppData()". Estos datos específicos de la aplicación podrían ser descripciones de eventos en un formato privado de aplicación o una clave de una base de datos de información de la aplicación.

Cuando se crea una grabación, las aplicaciones especifican un conjunto de propiedades de la grabación. Una de ellas es el periodo de expiración medido desde que se realizó la grabación. Tal como se define en la cláusula 7.3 anterior, una vez vencido dicho periodo, la implementación borra la grabación. Una segunda propiedad es un periodo de validez medido desde que se realizó la petición de la grabación. Si la grabación no se ha realizado antes de que venza dicho periodo de validez, la grabación puede ser suprimida.

Las aplicaciones pueden adquirir un conjunto de grabaciones representadas en una RecordingList. Las aplicaciones pueden desear limitar el conjunto de grabaciones de una RecordingList utilizando para ello un RecordingListFilter. En esta Recomendación se definen algunos filtros que están disponibles para las aplicaciones. Las especificaciones de grabación GEM pueden definir sus propios filtros. Las aplicaciones pueden crear sus propios filtros. Los filtros pueden aplicarse en tándem de forma que la salida de un filtro constituye la entrada para el filtro siguiente.

NOTA – Las especificaciones de grabación GEM pueden definir reglas que rijan el acceso de las aplicaciones a las grabaciones. Cuando así ocurre, la API de enumeración no devuelve las grabaciones a las que la aplicación llamante no tenga derecho de acceso de acuerdo con dichas reglas.

Cuando se comienzan a grabar los datos de una grabación, se crea un RecordedService. Dicho RecordedService amplía la interfaz del servicio JavaTV y constituye el enlace entre la API de grabación y la API de reproducción.

8.1.2 Información detallada

Se soportan los paquetes org.ocap.shared.dvr y org.ocap.shared.dvr.navigation.

En ServiceContextRecordingSpec, las especificaciones de grabación GEM puede hacer que sea obligatoria la que en otro caso sería característica opcional de almacenamiento y grabación de contenidos de la memoria de desplazamiento temporal cuando el startTime (tiempo de inicio) ya ha pasado.

Si una grabación programada comienza cuando el dispositivo terminal de grabación GEM está apagado, pero se enciende mientras la grabación programada aún está vigente, la implementación dará los pasos siguientes:

1) Fija el estado de LeafRecordingRequest de acuerdo con los recursos disponibles.
2) Si existen recursos disponibles, crea un RecordedService e inicia una grabación asociada.
3) Si la grabación se completa, fija el estado de LeafRecordingRequest en INCOMPLETE_STATE.
4) Fija que la excepción devuelta por el método LeafRecordingRequest.getFailedException sea org.ocap.shared.dvr.RecordingFailedException.POWER_INTERRUPTION.

Cuando una grabación programada está en curso y se suprime la alimentación de energía del terminal de grabación GEM, pero el dispositivo vuelve a alimentarse y aún no ha terminado la grabación del contenido en cuestión, la implementación ejecutará los pasos siguientes:

1) Crea una LeafRecordingRequest para el resto de la grabación tan pronto como finalice el proceso de arranque y la implementación detecta que una grabación programada debiera estar en curso. Copia los datos específicos de la aplicación de la LeafRecordingRequest original a la que ha creado. Fija el estado de acuerdo con los recursos disponibles. Fija RecordingSpec para que concuerde con la petición de grabación original.
2) Si existen recursos disponibles, crea un RecordedService e inicia una grabación asociada.
3) Si la grabación se completa, fija el estado de LeafRecordingRequest en INCOMPLETE_STATE.
4) Fija que la excepción devuelta por el método LeafRecordingRequest.getFailedException sea org.ocap.shared.dvr.RecordingFailedException.POWER_INTERRUPTION.

Si la alimentación se suprime y vuelve a aplicarse varias veces al terminal de grabación GEM mientras dura una grabación programada, se crean una LeafRecordingRequest y una RecordedService cada vez que la alimentación vuelve a aplicarse y los recursos quedan disponibles para comenzar la grabación.

NOTA 1 – Las especificaciones de grabación GEM pueden añadir requerimientos adicionales para combinar partes de una grabación interrumpida por la pérdida de la alimentación de energía en una única construcción (por ejemplo, una subclase de RecordingRequest), permitiendo así que sean reproducidas o manipuladas como una única entidad.

Cuando una grabación programada está en curso y se suprime la alimentación de energía del terminal de grabación GEM, pero el dispositivo vuelve a alimentarse cuando la grabación del contenido en cuestión ya ha terminado, la implementación ejecutará los pasos siguientes:

1) Fija el estado de LeafRecordingRequest en INCOMPLETE_STATE.
2) Fija que la excepción devuelta por el método LeafRecordingRequest.getFaileException sea INSUFFICIENT_RESOURCES.

Cuando una grabación programada está en curso y se suprime la alimentación de energía del terminal de grabación GEM, pero el dispositivo vuelve a alimentarse cuando aún NO ha terminado la grabación del contenido en cuestión, la implementación ejecutará los pasos siguientes:

1) Crea una LeafRecordingRequest para el resto de la grabación tan pronto como finaliza el proceso de arranque y la implementación detecta que una grabación programada debiera estar en curso. Copia los datos específicos de la aplicación de la LeafRecordingRequest original a la recién creada. Fija el estado de acuerdo con los recursos disponibles. Fija RecordingSpec para que concuerde con la petición de grabación original.
2) Si existen recursos disponibles, crea un RecordedService e inicia una grabación asociada.
3) Si la grabación se completa, fija el estado de LeafRecordingRequest en INCOMPLETE_STATE.
4) Fija que la excepción devuelta por el método LeafRecordingRequest.getFailedException sea org.ocap.shared.dvr.RecordingFailedException.POWER_INTERRUPTION.

Si la alimentación de suprime y vuelve a aplicarse varias veces al terminal de grabación GEM mientras dura una grabación programada, se crean una LeafRecordingRequest y una RecordedService cada vez que vuelva a aplicarse la alimentación y los recursos quedan disponibles para comenzar la grabación.

NOTA 2 – Las especificaciones de grabación GEM pueden añadir requerimientos adicionales para combinar partes de una grabación interrumpida por la pérdida de la alimentación de energía en una única construcción (por ejemplo, una subclase de RecordingRequest), permitiendo así que sean reproducidas o manipuladas como una única entidad.

8.2 Reproducción

8.2.1 Visión general (informativa)

La reproducción de una grabación puede hacerse de dos formas distintas en función de si se desea o no iniciar aplicaciones que puedan formar parte de la grabación.

1) Haciendo una llamada al método select(Service) en un ServiceContext del RecordedService que ha de reproducirse, se seleccionan todos los componentes de servicio del servicio en cuestión, incluidas cualesquiera aplicaciones de inicio que hayan sido grabadas.

2) Haciendo una llamada al método getMediaLocator() en un RecordedService y pasando el resultado a javax.media.Manager.createPlayer(MediaLocator). Una vez obtenido el reproductor (Player), start() o syncStart() iniciarán la reproducción. Ello hace que sólo se reproduzcan los flujos grabables de la grabación y no cualquier otra aplicación.

Una vez que ha empezado la reproducción, algunas de las formas de controlarla son las siguientes:

1) Utilizar los controles de JMF devueltos por Player.getControl(String) o Player.getControls() para características tales como la selección de idioma de audio y el escalado del vídeo.

2) Utilizar Player.setRate(float) para controlar la velocidad de reproducción cambiando entre normal, rápida, avance rápido, retroceso rápido y pausa.

8.2.2 Información detallada

Se soportará el paquete org.ocap.shared.media. Los reproductores JMF que presentan el contenido de una memoria intermedia de desplazamiento temporal soportarán el control TimeshiftControl de dicho paquete.

Las ampliaciones siguientes se aplicarán a las API requeridas por [ETSI TS 102 819]:

1) El método javax.tv.service.selection.ServiceContext.select(Service) aceptará instancias de RecordedService como entradas válidas. Cuando se le llama con dicha instancia, el contenido grabado de dicho RecordedService se seleccionará en el ServiceContext especificado.

2) El método javax.media.Manager.createPlayer(MediaLocator) aceptará los mediaLocators devueltos por RecordedService.getMediaLocator como entradas válidas y devuelve un reproductor JMF. Cuando dicho reproductor pasa al estado de iniciado, se presentan los flujos grabables de dicho RecordedService.

3) Cuando se selecciona con éxito un RecordedService en un ServiceContext, las AppsDatabase se rellenan utilizando el conjunto de aplicaciones grabadas como parte de dicho RecordedService, como si dichas aplicaciones se transmitieran en directo.

   NOTA 1 – La AppsDatabase debería actualizarse en la medida en que la especificación de grabación GEM requiera cambios en la supervisión y reconstrucción de la señalización de la aplicación.

4) Si durante la reproducción de flujos grabables (tanto la reproducción de una grabación programada como una grabación con desplazamiento temporal) se produce un cambio de la velocidad de reproducción, se enviará un javax.media.RateChangeEvent a todas las aplicaciones con ControllerListeners registradas en un reproductor JMF para dicho contenido.

5) Las llamadas al método setRate de un reproductor JMF controlarán la velocidad de reproducción, incluidos los cambios entre velocidad normal, avance rápido, retroceso rápido y pausa.

6) Las llamadas al método Player.setMediaTime tratarán de iniciar la presentación de contenidos tan cerca como sea posible del tiempo de medios especificado, salvo cuando se pase un tiempo de medios que haya sido devuelto por MediaTimeFactoryControl. Se utiliza SetTimeApproximations cuando se respete la semántica definida por dicho método.

7) Cuando se reproduzca un RecordedService, las llamadas al método ServiceDomain.attach(..) con el localizador (Locator) devuelto por RecordedService.getLocator funcionarán tal como se especifica y proporciona acceso al sistema de ficheros de difusión en la grabación. En esta Recomendación no se definen los requisitos de acceso a los sistemas de ficheros de difusión en una grabación cuando ésta no se reproduce, sin embargo, dichos requisitos pueden definirse en las especificaciones de grabación GEM.

Cuando los reproductores JMF están presentando los flujos grabables de un RecordedService o el contenido de una memoria intermedia de desplazamiento temporal, se soportarán como mínimo los controles JMF siguientes (definidos por [ETSI TS 102 819], por esta Recomendación o especificados de forma explícita):

- org.davic.media.AudioLanguageControl
- org.davic.media.FreezeControl
- javax.tv.media.MediaSelectControl
- org.ocap.shared.media.TimeLineControl
- org.ocap.shared.media.TimeFactoryControl
- org.davic.media.MediaTimeEventControl tal como se define en [b-DAVIC 1.4.1p9].

NOTA 2 – Las especificaciones de grabación GEM pueden requerir que se soporten controles JMF adicionales para RecordedServices o los contenidos de una memoria intermedia de desplazamiento temporal. Para ambos casos pueden especificarse conjuntos de controles JMF diferentes.

8.3 Otras API

8.3.1 Determinación de la versión

Las propiedades enumeradas en los dos cuadros siguientes se incluirán en el conjunto de propiedades de la clase java.lang.System. Por tanto, dichas propiedades pueden recuperarse utilizando java.lang.System.getProperty(). Dado que esta API devuelve una cadena, los valores numéricos devueltos se codificarán tal como define java.lang.Integer.toString(int).

Cuadro 8-1 – Propiedades del sistema para indagar sobre la versión

Propiedad	Semántica	Posibles valores	Ejemplo
gem.recording.version.major	Número de versión principal de la versión de la Recomendación actual soportado.	Valor entero no negativo	"1"
gem.recording.version.minor	Número de versión menor de la versión de la Recomedación actual soportado.	Valor entero no negativo	"0"
gem.recording.version.micro	Número de versión micro de la versión de la Recomendación actual soportado.	Valor entero no negativo	"0"

8.4 Permisos

8.4.1 Aplicaciones sin signo

De entre las RecordingPermission, sólo se concederá RecordingPermission ("read", "own") a aplicaciones sin signo.

8.4.2 Aplicaciones con signo

Cuando el fichero de peticiones de permiso solicita permiso para crear una petición de grabación y éste se concede, se crea un RecordingPermission("create", "own").

Cuando el fichero de peticiones de permiso solicita permiso para modificar una petición de grabación y éste se concede, se crea un RecordingPermission("modify", "own").

Cuando el fichero de peticiones de permiso solicita permiso para suprimir una petición de grabación y éste se concede, se crea un RecordingPermission("delete", "own").

Cuando el fichero de peticiones de permiso solicita permiso para cancelar una petición de grabación y éste se concede, se crea un RecordingPermission("cancel", "own").

Las especificaciones de grabación GEM pueden definir un mecanismo que permita que las aplicaciones tengan instancias de RecordingPermission cuya cadena de acción sea "*", pero no constituye un requisito.

9 Señalización de la aplicación

9.1 Descripción de la grabación de aplicaciones

Las especificaciones de grabación GEM definirán una descripción de grabación de aplicación suficiente para que de ella se desprenda lo siguiente:

Cuadro 9-1 – Descripción de grabación de aplicación

Función	Tipo
scheduled_recording__flag	booleano
Trick_mode_aware_flag	booleano
Time_shift_flag	booleano
initiating_replay_flag	booleano
label_count	entero sin signo
for(i=0;i<N0;i++){	
label_length	entero sin signo de 8 bits
for(j=0;i<N1;i++){	
label_payload	entero sin signo de 8 bits
}	
storage_properties	entero sin signo de 2 bits
}	

scheduled_recording_flag (*bandera de grabación programada*): cuando esta bandera de un único bit se pone a '1', indica que la aplicación es adecuada para grabar cuando el servicio en el que se señaliza se graba mediante una grabación programada. Cuando se pone a '0', indica que la aplicación no es adecuada para grabar mediante una grabación planificada. Son ejemplos de por qué una aplicación resultaría inadecuada para la grabación tanto una aplicación que no haya sido probada en un entorno PDR como una aplicación que esté estrechamente relacionada con el tiempo en el que se realiza de transmisión y carezca de significado para el usuario final si se reproduce a partir de una grabación (por ejemplo, una aplicación ligada a un evento en directo).

trick_mode_aware_flag (*bandera de reconocimiento del modo truco*): cuando esta bandera de un único bit se pone a '1', indica que la aplicación conoce que funciona en modo truco (*trick-mode*). Cuando se pone a '0', indica que la aplicación no es consciente del funcionamiento en modo truco.

time_shift_flag (*bandera de desplazamiento temporal*): cuando esta bandera de un único bit se pone a '1', indica que la aplicación es adecuada para grabar cuando el servicio en el que se señaliza se graba en el modo de grabación con desplazamiento temporal. Cuando se pone a '0', indica que la aplicación no es adecuada para grabar en el modo de grabación con desplazamiento temporal.

initiating_replay_flag (*bandera de inicio de reproducción*): cuando esta bandera de un único bit se pone a '1', indica que el terminal de grabación GEM no inicia la reproducción de flujos grabables en la misma grabación que la aplicación. La aplicación es responsable de iniciar dicha reproducción. Cuando se pone a '0', la implementación iniciará esta reproducción en paralelo con el comienzo de la aplicación, tal como sería el caso convencional.

NOTA 1 – Si se graban varias aplicaciones con un programa o un servicio y dichas aplicaciones se señalizan como que constituyen el punto de entrada del programa de reproducción, cuando se reproducen los flujos grabables conexos se arranca la primera aplicación de auto inicio conexa que se encuentra en el AIT almacenado.

label_count (*cómputo de etiquetas*): este campo de 8 bits identifica el número de etiquetas que se han utilizado.

label_length (*longitud de la etiqueta*): este campo de 8 bits identifica el número de bytes de la etiqueta.

label_char (*caracteres de la etiqueta*): este campo de 8 bits transporta un conjunto de bytes que etiquetan una parte de la aplicación en su protocolo de transporte.

NOTA 2 – Esta Recomendación no define qué partes de las aplicaciones pueden ser etiquetadas o la forma de la etiqueta (si la hay). El etiquetado puede hacerse a nivel de ficheros o de grupos de ficheros o a algún nivel inferior específico del protocolo utilizado para transportar la aplicación.

storage_properties (*propiedades de almacenamiento*): un campo que indica la importancia del almacenamiento de la parte etiquetada de la aplicación. Los valores y significados de este campo son los siguientes:

0 no debería almacenarse

1 el almacenamiento es crítico

2 el almacenamiento es opcional

3 reservado

Las especificaciones de grabación GEM que incluyen la definición MHP del equivalente funcional de la "señalización de aplicación" GEM satisfarán este requisito soportando el descriptor de grabación de aplicación definido en el anexo A.

10 Modelo de las aplicaciones

10.1 Ciclo de vida de la aplicación y reproducción en modo truco (*trick-mode*)

Cuando se ha iniciado la reproducción de contenidos grabados mediante la selección de un RecordedService, el modelo de aplicación y el ciclo de vida se modificarán de la forma siguiente:

1) La ejecución de aplicaciones no expresamente identificadas como conscientes del modo truco (la bandera trick_mode_aware_flag de la descripción de la grabación de la aplicación está a '1') se detiene cuando la reproducción cambia de normal al modo truco. Las aplicaciones expresamente identificadas como conscientes del modo truco siguen ejecutándose.

2) Cuando la reproducción abandona el modo truco y vuelve al modo normal, el terminal de grabación GEM evaluará la señalización de la aplicación del contenido para dicho punto, y arranca o detiene las aplicaciones según sea necesario. Las aplicaciones que se arrancan lo harán como si el usuario hubiera pasado a la difusión del contenido en cuestión.

NOTA – Las especificaciones de grabación GEM pueden permitir un cierto retardo en el reinicio de las aplicaciones después de volver a una reproducción normal si se considera que ello mejora la experiencia del usuario, por ejemplo, durante los ciclos repetidos de avance rápido/reproducción/avance rápido/reproducción.

Las transiciones desde reproducción en directo a reproducción con desplazamiento temporal no detendrán automáticamente las aplicaciones MHP mientras el contenido con desplazamiento temporal se esté reproduciendo a partir del final de la memoria intermedia de desplazamiento temporal (es decir, el punto en el que se escribe el contenido recién grabado).

11 Seguridad

11.1 Introducción (informativa)

Esta Recomendación incluye 2 modelos de seguridad que rigen la capacidad de las aplicaciones para funcionar con peticiones de grabación y grabaciones finalizadas.

1) Un modelo se basa en asociar atributos con aplicaciones MHP/OCAP. Estos atributos se expresan como clase de permiso Java. Algunas llamadas a métodos están protegidas y generan una SecurityException (excepción de seguridad) cuando aplicaciones sin permisos especificados realizan las llamadas. Este modelo es independiente de la información detallada de la RecordingRequest en cuestión.
2) El segundo modelo se basa en asociar atributos de seguridad con peticiones de grabación individuales. Estos atributos determinan las operaciones que puede realizar cada aplicación en dicha petición de grabación. Esta Recomendación no hace referencia alguna al mecanismo mediante el que dichos atributos se asocian con una petición de grabación

En el caso normal, la capacidad que tiene una aplicación para utilizar las características incluidas en esta Recomendación se rige por la intersección de ambos conjuntos de atributos. Las operaciones de una RecordingRequest fracasarán salvo que la aplicación llamante tenga los permisos Java necesarios para la operación y los atributos asociados con la RecordingRequest sean aplicados una vez que se permita que la aplicación llamante realice dicha operación. Las especificaciones de grabación GEM pueden definir un mecanismo para que aplicaciones que gocen de una gran confianza obtengan un permiso que les habilite para evitar el segundo modelo y tengan el derecho a realizar todas las operaciones en todas las RecordingRequests.

11.2 Fichero de petición de permiso

El DTD del fichero de petición de permiso definido por la especificación de terminal GEM incluirá al menos el elemento siguiente y los atributos asociados:

```
<!ELEMENT recordingpermission EMPTY>
<!ATTLIST recordingpermission
create (true|false) "false"
modify (true|false) "false"
delete (true|false) "false"
cancel (true|false) "false"
>
```

12 Requisitos mínimos del receptor

Los requisitos siguientes se aplicarán a todos los terminales de grabación GEM:

1) Se soportarán al menos las velocidades de reproducción variable siguientes: –16x, –8x, –4x, –2x, –1x, 0, 0,5x, 1x, 2x, 4x, 8x, 16x..

NOTA – para la velocidad de –1x, se permite que solamente se reproduzcan i-tramas.

Anexo A

Descripción de la grabación de aplicaciones

Las especificaciones de grabación GEM que incluyen la definición MHP del equivalente funcional de "señalización de aplicación" GEM, cumplirán con este requisito merced a soportar el descriptor de grabación de aplicaciones definido tal como se indica a continuación.

El descriptor de grabación de aplicación puede ser señalizado en el bucle descriptor de aplicación del AIT. Este descriptor contiene información adicional sobre el ciclo de vida de la aplicación, indicando en particular si una aplicación es adecuada para ser utilizada en las condiciones de reproducción del modo truco. Indica si esta aplicación va a ser o no grabada y cuándo se graba un programa, junto con el cual se señaliza dicha aplicación. Proporciona una forma de especificar las ubicaciones de los recursos de datos que se grabarán junto con la aplicación, así como las etiquetas de los módulos de carrusel objeto de la aplicación que deberán, deberían o no deberían ser grabados.

Cuadro A.1 – Sintaxis del descriptor de grabación de aplicación

Sintaxis	N.° de bits	Identificador	Comentarios/Valor
application_recording_descriptor (){			
descriptor_tag	8	uimsbf	6
descriptor_length	8	uimsbf	
scheduled_recording_flag	1	bslbf	
trick_mode_aware_flag	1	bslbf	
time_shift_flag	1	bslbf	
dynamic_flag	1	bslbf	
av_synced_flag	1	bslbf	
initiating_replay_flag	1	bslbf	
Reserved	7	bslbf	
label_count	8	uimsbf	N0
for(i=0;i<N0;i++){			
label_length	8	uimsbf	N1
for(j=0; j<N1; j++) {			
label_char	8	uimsbf	
}			
storage_properties	2	uimsbf	
Reserved	6		
}			
component_tag_list_length	8	uimsbf	N2
for(i=0;i<N2;i++){			
component_tag	8	uimsbf	
}			
private_length	8	uimsbf	N3
for(i=0;i<N3;i++){			
Private	8	uimsbf	
}			
for(i=0;i<N4;i++){			
reserved_future_use	8	uimsbf	
}			
}			

La semántica de los campos definidos en este descriptor será tal como se define en la cláusula 9.1, salvo que en esta cláusula se especifique otra cosa.

descriptor_tag (*etiqueta de descriptor*): entero de 8 bits con valor 0x06 que identifica al descriptor.

dynamic_flag (*bandera de dinámica*): bandera que indica si la aplicación se basa en la utilización de datos dinámicos durante su ejecución. Cuando se pone a 1, indica que la aplicación se basa en la presencia de ficheros (ya sean de código o de datos) o de señalización de aplicación (por ejemplo, código de control de aplicación) que cambia durante la vida del contenido en cuestión.

NOTA 1 – Esta Recomendación no define un comportamiento de terminales de grabación GEM que esté condicionado por el valor de esta bandera. Las especificaciones de terminal GEM pueden utilizar esta bandera en su determinación de si una aplicación es o no es grabable.

av_synced_flag (*bandera sincronizada*): esta bandera indica si la aplicación requiere la utilización de eventos de disparo. Si lo requiere, toma el valor '1'.

NOTA 2 – Esta Recomendación no define un comportamiento de terminales de grabación GEM que esté condicionado por el valor de esta bandera. Las especificaciones de terminal GEM pueden utilizar esta bandera en su determinación de si una aplicación es o no es grabable.

label_char (*caracteres de la etiqueta*): este campo de 8 bits transporta un conjunto de bits que constituyen una etiqueta de módulo. Esta etiqueta se corresponde con una etiqueta de uno o más módulos transportados por descriptores de etiqueta en los campos userInfo de la estructura moduleInfo de los DII, tal como se define en la cláusula B.2.2.4.1 de [ETSI ES 201 812] y [ETSI TS 102 812].

component_tag_list_length (*longitud de lista de etiqueta de componente*): este entero especifica la longitud expresada en número de bytes de la lista de etiquetas de componentes.

component_tag (*etiqueta de componente*): este campo identifica una componente de servicio que entrega datos, que la aplicación necesita en el momento de la reproducción y que se grabarán junto con la aplicación. Los componentes que transportan flujos grabables sólo deben ser incluidos en esta lista en caso de que deban grabarse componentes distintos de los componentes por defecto. Los componentes que son flujos que transportan carruseles de objetos DSMCC o tablas de información de aplicación MHP no deberían ser incluidos en la misma, ignorándose en caso de haberlo sido. Excepto esos casos, los componentes que sean flujos que transporten secciones privadas MPEG-2 deberían ser incluidos en la misma en caso de que se deseen grabar.

private (*privado*): estos bytes pueden utilizarse para ampliaciones privadas.

reserved_future_use (*reservado para uso futuro*): estos bytes reservados pueden ser utilizado para ampliaciones futuras de DVB.

Anexo B

Responsabilidades de las especificaciones de grabación GEM

B.1 Requeridas

A continuación se presenta una lista de elementos que en esta Recomendación se consideran fuera del ámbito del mismo y que deben incluirse en las especificaciones de grabación GEM.

1) Qué tipos de flujos deben considerarse "flujos grabables". Los tipos de flujos correspondientes a las cláusulas 7.2.1 ("Audio") y 7.2.2 ("Vídeo") de GEM en los que se basa la especificación de grabación GEM deben considerarse "flujos grabables".

2) Mecanismos para resolver los conflictos entre grabaciones solicitadas (por ejemplo, utilización de sintonizador).

3) Capacidades mínimas relativas al número de flujos (o número de flujos de cada tipo) que un terminal de grabación GEM puede grabar.

4) Definición de qué aplicaciones son grabables en grabación programada y con desplazamiento temporal (no tienen que ser necesariamente las mismas).

5) Requisitos de un terminal de grabación GEM para supervisar datos dinámicos (en el carrusel de objetos DSMCC o su equivalente funcional GEM) durante una grabación programada y con desplazamiento temporal (no tienen que ser necesariamente los mismos).

6) Requisitos de un terminal de grabación GEM para supervisar factores de activación GEM o eventos de flujo DSMCC durante grabación programada y con desplazamiento temporal (que tienen que ser necesariamente los mismos).

7) Requisitos de un terminal de grabación GEM para supervisar la señalización de aplicaciones dinámicas durante grabación programada y con desplazamiento temporal (que tienen que ser necesariamente los mismos).

8) Requisitos relativos a la reconstrucción de la temporización de datos dinámicos, eventos de activación/flujos y señalización de aplicaciones dinámicas durante grabaciones programadas y con desplazamiento temporal (no tienen que ser necesariamente los mismos). Las especificaciones de grabación GEM deben definir requisitos para la reproducción normal y en modo truco.

9) La precisión del periodo de expiración impuesto por las aplicaciones.

10) La definición de al menos un protocolo para las secuencias temporales transmitidas.

11) Las condiciones en las que un reproductor JMF o un contexto de servicio tengan una memoria intermedia de desplazamiento temporal adjunta.

12) Requisitos relativos al tamaño de los datos privados específicos de la aplicación que es posible asociar con una clave única sin que se genere una IllegalArgumentException.

13) Requisitos relativos al número de entradas de datos privados específicos de la aplicación que es posible asociar con una RecordingRequest única sin que se genere una NoMoreDataEntriesException.

14) Un mecanismo parta asociar atributos de seguridad con peticiones de grabación individuales y una correspondencia entre dicho mecanismo y el lenguaje en cada uno de los métodos en el cuadro siguiente relativo a "atributos de seguridad específicos de RecordingRequest (petición de grabación)".

Cuadro B.1 – Métodos con dependencias de los atributos de seguridad específicos de una petición de grabación

Método
`RecordingManager.addRecordingChangedListener(RecordingListListener)`
`RecordingManager.getEntries()`
`RecordingManager.getEntries(RecordingListFilter)`
`RecordingRequest.getRecordingProperties(int)`
`RecordingRequest.add.AppData(int,java.io.Serializable)`
`RecordingRequest.removeAppData(int)`
`RecordingRequest.setRecordingProperties(RecordingSpec)`
`RecordingRequest.delete()`
`RecordingManager.record()`
`RecordingRequest.cancel()`
`RecordingRequest.stop()`
`LeafRecordingRequest.getService()`
`RecordedService.setMediaTime()`

B.2 Opcionales

A continuación se presenta una lista de elementos que en esta Recomendación se consideran fuera del ámbito de la misma y que pueden incluirse en las especificaciones de grabación GEM.

1) Mecanismos para controlar en qué medida una aplicación puede leer o modificar grabaciones programadas y grabaciones finalizadas realizadas por otra aplicación.

2) Subclases de RecordingListFilter para filtrar la lista de grabaciones de maneras no soportadas por esta Recomendación.

3) Reglas para determinar las grabaciones a las que puede acceder una aplicación.

4) Controles JMF adicionales que deben soportarse para RecordedServices o los contenidos de una memoria intermedia de desplazamiento temporal. Para ambos casos pueden especificarse distintos conjuntos de controles JMF.

5) Retardos en la reinicialización de aplicaciones después de volver a la reproducción normal si con ello se considera que se mejora la experiencia del usuario, por ejemplo, cuando se producen ciclos repetidos de avance rápido/reproducción normal/avance rápido/reproducción normal.

6) Un mecanismo para permitir aplicaciones de gran confianza para conseguir instancias de RecordingPermission cuyo parámetro de acción sea "*".

7) Que el comportamiento opcional definido en la descripción de clase de ServiceContextRecordingSpec, en el que se almacenan los contenidos de la memoria intermedia de desplazamiento temporal cuando el parámetro startTime corresponda a un momento pasado, se convierta en obligatorio en esa especificación concreta de grabación GEM.

8) Requisitos relativos a la reconstrucción de la temporización de datos dinámicos, eventos de activación/flujos y señalización de aplicaciones dinámicas durante grabaciones programadas y con desplazamiento temporal (tienen que ser necesariamente los mismos). Las especificaciones de grabación GEM deben definir requisitos para la reproducción normal y en modo truco.

9) La precisión del periodo de expiración impuesto por las aplicaciones.

10) La definición de al menos un protocolo para las secuencias temporales transmitidas.

11) Las condiciones en las que un reproductor JMF o un contexto de servicio tengan una memoria intermedia de desplazamiento temporal adjunta.

12) Requisitos relativos al tamaño de los datos privados específicos de la aplicación que es posible asociar con una clave única sin que se genere una IllegalArgumentException.

13) Requisitos relativos al número de entradas de datos privados específicos de la aplicación que es posible asociar con una RecordingRequest única sin que se genere una NoMoreDataEntriesException.

14) Un mecanismo para asociar atributos de seguridad con peticiones de grabación individuales y una correspondencia entre dicho mecanismo y el lenguaje en cada uno de los métodos en el cuadro B.1 relativo a "atributos de seguridad específicos de RecordingRequest (petición de grabación)".

Anexo C

Referencias externas; erratas, clarificaciones y excepciones

C.1 Marco de medios Java (JMF, *Java Media Framework*)

C.1.1 javax.media.Clock

En esta Recomendación se pretende clarificar, tal como se indica a continuación, los textos siguientes que están incluidos en la especificación para esta clase.

La transformación que define un reloj con una base de tiempos (TimeBase) viene dada por tres parámetros: velocidad, instante de arranque de medios (mst, media starttime), y tiempo de inicio de base de tiempo (tbst, *time-base start-time*). Dado un tiempo de base de tiempo (tbt, *time-base time*), el tiempo de medios (mt, *media time*) puede calcularse mediante la transformación siguiente:

$$mt = mst + (tbt – tbst) * velocidad$$

La velocidad es simplemente un factor de escala que se aplica a la base de tiempo (TimeBase). Por ejemplo, una velocidad de 2,0 indica que el reloj avanza al doble de la velocidad de su TimeBase Igualmente, una velocidad negativa indica que el reloj avanza en sentido contrario a su TimeBase.

El tiempo de inicio de base de tiempo y el tiempo de inicio de medios definen un punto común en el tiempo en el que el reloj y la base de tiempo (TimeBase) están sincronizados.

Cada cambio exitoso de velocidad se considerará que es "un punto común en el tiempo en el que el reloj y la base de tiempo (TimeBase) están sincronizados", tal como se define más arriba. Los valores del reloj y de la base de tiempo en el punto común serán los valores en el momento del cambio de velocidad.

Anexo D

Paquetes API de núcleo común de una plataforma de registro vídeo numérico

D.1 Paquete de registro vídeo numérico compartido

```
org.ocap.shared.dvr
     Package
```


D.1.1 Clase ServiceRecordingSpec

```
org.ocap.shared.dvr
     Class ServiceRecordingSpec

java.lang.Object
  +--org.ocap.shared.dvr.RecordingSpec
       +--org.ocap.shared.dvr.ServiceRecordingSpec
```

---

clase pública ServiceRecordingSpec

amplía RecordingSpec

Especifica una petición de grabación en términos de un servicio.

Cuando instancias de esta clase se pasan a RecordingManager.record(..), serán de aplicación los modos de fallo siguientes, si el tiempo de finalización (calculado como tiempo de inicio + duración)

ya ha pasado cuando se hace la llamada al método de grabación, en cuyo caso el método de grabación generará una IllegalArgumentException.

Cuando una instancia de esta especificación de grabación se pasa como parámetro al método RecordingRequest.reschedule(..), se genera una IllegalArgumentException si la fuente es distinta de la que se especifica en la especificación de grabación actual para la petición de grabación y si ésta se encuentra en curso.

Cuando instancias de esta clase se pasan a RecordingManager.record(..), si el tiempo de inicio ya ha pasado y

– ningún contenido concernido ha sido ya grabado;

– parte del contenido concernido ha sido ya grabado pero la implementación no permite incluir contenido grabado en una grabación programada,

entonces, se utilizará el tiempo actual como tiempo de inicio y la duración se reduce en consecuencia. Esta Recomendación no exige que las implementaciones incluyan contenidos ya grabados en grabaciones programadas; sin embargo, las especificaciones de grabaciones GEM sí pueden exigirlo.

Resumen del constructor
`ServiceRecordingSpec(javax.tv.service.Service source, java.util.Date startTime, long duration, RecordingProperties properties)` **Constructor**

Resumen del método	
`long`	**`getDuration()`** Devuelve la duración pasada como argumento al constructor.
`javax.tv.service.Service`	**`getSource()`** Devuelve la fuente de la grabación
`java.util.Date`	**`getStartTime()`** Devuelve el tiempo de inicio pasado como argumento al constructor.

Métodos heredados de la clase org.ocap.shared.dvr.RecordingSpec
`getProperties`

Métodos heredados de la clase java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Información detallada del constructor

ServiceRecordingSpec

```
public ServiceRecordingSpec(javax.tv.service.Service source,
                            java.util.Date startTime,
                            long duration,
                            RecordingProperties properties)
                     throws java.lang.IllegalArgumentException
```

Constructor.

Parámetros:

`source` – servicio que debe grabarse

`startTime` – tiempo de inicio de la grabación

`duration` – duración de la grabación en milisegundos

`properties` – definición de cómo debe hacerse la grabación.

Genera:

`java.lang.IllegalArgumentException` – si la fuente no es un servicio de difusión o si la duración es negativa

Información detallada del método

getSource

```
public javax.tv.service.Service getSource()
```

Devuelve la fuente de la grabación.

Devuelve:

La fuente pasada al contructor.

---

getStartTime

```
public java.util.Date getStartTime()
```

Devuelve el tiempo de inicio pasado como argumento al constructor.

Devuelve:

El tiempo de inicio pasado al constructor.

---

getDuration

```
public long getDuration()
```

Devuelve la duración pasada como argumento al constructor.

Devuelve:

La duración pasada al constructor.

D.1.2 Clase AccessDeniedException

```
org.ocap.shared.dvr
     Class AccessDeniedException

java.lang.Object
  +--java.lang.Throwable
       +--java.lang.Exception
            +--org.ocap.shared.dvr.AccessDeniedException
```

Todas las interfaces implementadas:

java.io.Serializable

---

clase pública **AccessDeniedException**

amplía java.lang.Exception

Excepción generada cuando una aplicación se bloquea para que no pueda funcionar con una RecordingRequest mediante atributos de seguridad asociados con dicha RecordingRequest.

Véase también:

Serialized Form

Resumen del constructor
`AccessDeniedException()` construye una AccessDeniedException sin mensaje de información detallada

Métodos heredados de la clase java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Métodos heredados de la clase java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Información detallada del constructor

AccessDeniedException

```
public AccessDeniedException()
```

Construye una AccessDeniedException sin mensaje de información detallada

D.1.3 Clase DeletionDetails

```
org.ocap.shared.dvr
     Class DeletionDetails

java.lang.Object
  +--org.ocap.shared.dvr.DeletionDetails
```

---

clase pública **DeletionDetails**

amplía java.lang.Object

Esta clase contiene información detallada sobre la supresión de un servicio grabado. La implementación construye las instancias de esta clase que son devueltas a las aplicaciones desde el método `getDeletionDetails`.

Resumen del campo	
`static int`	**`EXPIRED`** Código de motivo: el servicio grabado ha sido suprimido por la implementación al haber expirado la petición de grabación.
`static int`	**`USER_DELETED`** Código de motivo: el servicio grabado ha sido explícitamente suprimido por la aplicación.

Resumen del constructor
`DeletionDetails(int reason, java.util.Date date)` Construye una DeletionDetails

Resumen del método	
`java.util.Date`	**`getDeletionTime`**`()` Obtiene la fecha y hora en que el servicio grabado ha sido suprimido.
`int`	**`getReason`**`()` Informa del motivo por el que el servicio grabado ha sido suprimido.

Métodos heredados de la clase java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Información detallada del campo

EXPIRED

```
public static final int EXPIRED
```

Código de motivo: el servicio grabado ha sido suprimido por la implementación al haber expirado la petición de grabación.

Véase también:

Constant Field Values

USER_DELETED

```
public static final int USER_DELETED
```

Código de motivo: el servicio grabado ha sido explícitamente suprimido por la aplicación.

Véase también:

Constant Field Values

Información detallada del constructor

DeletionDetails

```
public DeletionDetails(int reason,
                       java.util.Date date)
```

Construye una DeletionDetails

Parámetros:

`reason` – motivo por el que ha suprimido el servicio grabado

`date` – fecha y hora en que el servicio grabado ha sido suprimido

Información detallada del método

getReason

```
public int getReason()
```

Informa del motivo por el que se ha suprimido el servicio grabado. Es el valor pasado al constructor.

Devuelve:

Código del motivo por el que se ha suprimido el servicio grabado.

---

getDeletionTime

```
public java.util.Date getDeletionTime()
```

Obtiene la fecha y hora en que el servicio grabado ha sido suprimido. Es el valor pasado al constructor.

Devuelve:

Fecha y hora de la supresión.

D.1.4 Interfaz LeafRecordingRequest

```
org.ocap.shared.dvr
     Interface LeafRecordingRequest
```

Todas las superinterfaces:

RecordingRequest

---

interfaz pública **LeafRecordingRequest**

amplía RecordingRequest

Esta interfaz representa información correspondiente a una petición de grabación a nivel de hoja. La petición de grabación que representa esta interfaz corresponde con una petición de grabación que ha sido resuelta completamente en una única grabación.

Una petición de grabación a nivel de hoja puede estar pendiente (es decir, esperando que llegue el tiempo de inicio), en curso, completa, incompleta o fallida.

Cuando está pendiente, una petición de grabación puede entrar en conflicto por recursos disputados por otras grabaciones. Dichos conflictos debieran solucionarse antes del tiempo programado de inicio de la grabación, pues si no, es previsible que la petición de grabación pendiente sea fallida.

Resumen del campo	
`static int`	**`COMPLETED_STATE`** La grabación de esta petición de grabación se ha completado con éxito.
`static int`	**`DELETED_STATE`** El servicio grabado correspondiente a esta petición de grabación se ha suprimido.
`static int`	**`FAILED_STATE`** La petición de grabación ha resultado fallida.
`static int`	**`IN_PROGRESS_INSUFFICIENT_SPACE_STATE`** La grabación de esta petición de grabación se ha iniciado y está en curso, pero la implementación ha detectado que el espacio de almacenamiento puede no ser suficiente para completar la grabación.
`static int`	**`IN_PROGRESS_STATE`** La grabación de esta petición de grabación se ha iniciado y está en curso.
`static int`	**`INCOMPLETE_STATE`** La grabación de esta petición de grabación se ha iniciado pero se ha producido un fallo en el IN_PROGRESS_STATE antes de alcanzar el COMPLETED_STATE.
`static int`	**`PENDING_NO_CONFLICT_STATE`** La petición de grabación está pendiente. Se prevé que la grabación de esta petición se complete con éxito.
`static int`	**`PENDING_WITH_CONFLICT_STATE`** La petición de grabación no puede iniciarse debido a conflictos por recursos.

Resumen del método	
void	**cancel**() Cancela una petición de grabación pendiente.
DeletionDetails	**getDeletionDetails**() Obtiene información detallada sobre la supresión del servicio grabado correspondiente a esta petición de grabación.
java.lang.Exception	**getFailedException**() Obtiene la excepción que ha causado que la petición de grabación pase al estado FAILED_STATE, o INCOMPLETE_STATE.
RecordedService	getService() Devuelve el RecordedService correspondiente a la petición de grabación.
void	**stop**() Detiene la grabación de una petición de grabación en curso con independencia de la duración que ya ha sido grabada.

Métodos heredados de la interfaz org.ocap.shared.dvr.RecordingRequest
addAppData, delete, getAppData, getAppID, getId, getKeys, getParent, getRecordingSpec, getRoot, getState, isRoot, removeAppData, reschedule, setRecordingProperties

Información detallada del campo

PENDING_NO_CONFLICT_STATE

```
public static final int PENDING_NO_CONFLICT_STATE
```

La petición de grabación está pendiente. Se prevé que la grabación de esta petición se complete con éxito.

Véase también:

Constant Field Values

PENDING_WITH_CONFLICT_STATE

```
public static final int PENDING_WITH_CONFLICT_STATE
```

La petición de grabación no puede iniciarse debido a conflictos por recursos. La implementación ha detectado un conflicto por la hora programada de esta petición y la resolución actual del conflicto no permite iniciar con esta petición grabación.

Véase también:

Constant Field Values

IN_PROGRESS_STATE

`public static final int` **IN_PROGRESS_STATE**

La grabación de esta petición de grabación se ha iniciado y está en curso. Se prevé que la grabación se complete con éxito.

Véase también:

Constant Field Values

IN_PROGRESS_INSUFFICIENT_SPACE_STATE

`public static final int` **IN_PROGRESS_INSUFFICIENT_SPACE_STATE**

La grabación de esta petición de grabación se ha iniciado y está en curso, pero la implementación ha detectado que el espacio de almacenamiento puede no ser suficiente para completar la grabación. Esta situación puede producirse cuando la implementación detecta que el espacio de almacenamiento puede no ser suficiente para completar la petición de grabación y otras peticiones de grabación que estén en curso. No es previsible que la petición de grabación se complete satisfactoriamente. Una petición de grabación pasa a este estado desde IN_PROGRESS_STATE o PENDING_NO_CONFLICT_STATE. La implementación puede iniciar una petición de grabación estando en IN_PROGRESS_STATE en base a una estimación inicial del espacio requerido y pasar luego a IN_PROGRESS_INSUFFICIENT_SPACE_STATE cuando la implementación tenga información suficiente para hacer una estimación más precisa del espacio necesario. Si una implementación no puede detectar espacio insuficiente con antelación, la implementación puede iniciar la petición de grabación en IN_PROGRESS_STATE y pasar a FAILED_STATE cuando se produzca una situación de escasez de espacio. También es posible que una petición de grabación se inicie en IN_PROGRESS_INSUFFICIENT_SPACE_STATE y pase luego a IN_PROGRESS_STATE o COMPLETED_STATE. Esto puede ocurrir si las demás peticiones de grabación se suprimen después de haberse iniciado la grabación o si la implementación hace una mejor estimación del espacio necesario conforme progresa la grabación.

Véase también:

Constant Field Values

INCOMPLETE_STATE

`public static final int` **INCOMPLETE_STATE**

La grabación de esta petición de grabación se ha iniciado, pero ha resultado fallida estando en IN_PROGRESS_STATE antes de poder alcanzar COMPLETED_STATE. La petición de grabación (RecordingRequest) contendrá un `RecordedService` que puede ser reproducido, cualquiera que sea la duración de la grabación realizada.

Véase también:

Constant Field Values

COMPLETED_STATE

```
public static final int COMPLETED_STATE
```

La grabación de esta petición de grabación se ha completado con éxito.

Véase también:

Constant Field Values

FAILED_STATE

```
public static final int FAILED_STATE
```

La petición de grabación ha resultado fallida.

Véase también:

Constant Field Values

DELETED_STATE

```
public static final int DELETED_STATE
```

El servicio grabado correspondiente a esta petición de grabación se ha suprimido.

Véase también:

Constant Field Values

Información detallada del método

cancel

```
public void cancel()
            throws java.lang.IllegalStateException,
                   AccessDeniedException
```

Cancela una petición de grabación pendiente. La petición de grabación se suprimirá de la base de datos después de invocar satisfactoriamente este método. La cancelación de una petición de grabación puede resolver uno o más conflictos. En ese caso, algunas grabaciones pendientes con conflicto se cambiarían a pendientes sin conflicto.

Genera:

`AccessDeniedException` – si no se permite que la aplicación llamante realice esta operación mediante atributos de seguridad específicos de RecordingRequest.

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("cancel",..) o RecordingPermission("*",..)

`java.lang.IllegalStateException` – si el estado de la grabación no es PENDING_STATE_NO_CONFLICT_STATE o PENDING_WITH_CONFLICT_STATE.

---

stop

```
public void stop()
          throws java.lang.IllegalStateException,
                 AccessDeniedException
```

Detiene la grabación durante una petición de grabación en curso con independencia de cuál haya sido la duración grabada. Pasa la grabación al estado INCOMPLETE_STATE.

Genera:

`AccessDeniedException` – si no se permite que la aplicación llamante realice esta operación mediante atributos de seguridad específicos de RecordingRequest.

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("cancel",..) o RecordingPermission("*",..)

`java.lang.IllegalStateException` – si el estado de la grabación no es IN_PROGRESS_STATE, o IN_PROGRESS_INSUFFICIENT_SPACE_STATE.

---

getFailedException

```
public java.lang.Exception getFailedException()
                                        throws java.lang.IllegalStateException
```

Obtiene la excepción que causó que la petición de grabación pasara al estado `FAILED_STATE`, o `INCOMPLETE_STATE`.

Devuelve:

La excepción que causó el fallo. La excepción devuelta será una RecordingFailedException.

Genera:

`java.lang.IllegalStateException` – si la petición de grabación no está en el estado FAILED_STATE o INCOMPLETE_STATE.

---

getService

```
public RecordedService getService()
                            throws java.lang.IllegalStateException,
                                   AccessDeniedException
```

Devuelve el `RecordedService` correspondiente a la petición de grabación.

Devuelve:

El servicio grabado asociado a la petición de grabación.

Genera:

`java.lang.IllegalStateException` – si la petición de grabación no está en el estado INCOMPLETE_STATE, IN_PROGRESS_STATE, IN_PROGRESS_INSUFFICIENT_SPACE_STATE o COMPLETED_STATE.

`AccessDeniedException` – si no se permite que la aplicación llamante realice esta operación mediante atributos de seguridad específicos de RecordingRequest.

---

getDeletionDetails

```
public DeletionDetails getDeletionDetails()
                                    throws java.lang.IllegalStateException
```

Obtiene información detallada sobre la supresión del servicio grabado correspondiente a esta petición de grabación.

Devuelve:

Información detallada de la supresión para esta petición de grabación.

Genera:

`java.lang.IllegalStateException` – si la petición de grabación no está en el estado DELETED_STATE.

D.1.5 Clase LocatorRecordingSpec

```
org.ocap.shared.dvr
     Class LocatorRecordingSpec

java.lang.Object
  +--org.ocap.shared.dvr.RecordingSpec
       +--org.ocap.shared.dvr.LocatorRecordingSpec
```

---

clase pública **LocatorRecordingSpec**

amplía RecordingSpec

Especifica una petición de grabación en términos de uno o más localizadores.

Si la fuente contiene múltiples localizadores, todos ellos DEBEN formar parte del mismo servicio.

Cuando se pasan instancias de esta clase a RecordingManager.record(..), se aplicará el modo de fallo adicional siguiente – si el tiempo de finalización (calculado como tiempo de inicio + duración) es anterior al momento de hacer la llamada al método de grabación, dicho método de grabación generará una IllegalArgumentException.

Cuando se pasa una instancia de esta especificación de grabación como parámetro a RecordingRequest.reschedule(..), se genera una IllegalArgumentException si la fuente es distinta de la fuente especificada en la actual especificación de grabación para la petición de grabación y si dicha petición de grabación está en curso.

Cuando se pasan instancias de esta clase a RecordingManager.record(..), el tiempo de inicio es anterior y se cumple que

– ningún contenido concernido ha sido ya grabado, o bien,

– parte del contenido concernido ya ha sido grabado pero la implementación no soporta la inclusión de contenido ya grabado en una grabación programada,

entonces, el tiempo actual se utilizará como tiempo de inicio, reduciéndose consecuentemente la duración. Esta Recomendación no requiere que la implementación incluya contenidos ya grabados en grabaciones programadas; sin embargo, las especificaciones de grabaciones GEM sí pueden exigirlo.

Resumen del constructor
LocatorRecordingSpec(javax.tv.locator.Locator[] source, java.util.Date startTime, long duration, RecordingProperties properties) Constructor

Resumen del método	
long	**getDuration**() Devuelve la duración pasada como argumento al constructor.
javax.tv.locator.Locator[]	**getSource**() Devuelve la fuente de la grabación.
java.util.Date	**getStartTime**() Devuelve el tiempo de inicio pasado como argumento al constructor.

Métodos heredados de la clase org.ocap.shared.dvr.RecordingSpec
getProperties

Métodos heredados de la clase java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Información detallada del constructor

LocatorRecordingSpec

```
public LocatorRecordingSpec(javax.tv.locator.Locator[] source,
                            java.util.Date startTime,
                            long duration,
                            RecordingProperties properties)
                     throws
javax.tv.service.selection.InvalidServiceComponentException
```

Constructor

Parámetros:

`source` – fuente de flujos a grabar. Las implementaciones harán una copia de esta matriz antes de que se devuelva el constructor

`startTime` – tiempo de inicio de la grabación

`duration` – duración de la grabación en milisegundos

`properties` – definición de cómo ha de hacerse la grabación

Genera:

`javax.tv.service.selection.InvalidServiceComponentException` – si todos los localizadores del parámetro fuente no se encuentran en el mismo servicio.

`java.lang.IllegalArgumentException` – si la duración es negativa.

Información detallada del método

getSource

```
public javax.tv.locator.Locator[] getSource()
```

Devuelve la fuente de la grabación

Devuelve:

La fuente pasada al constructor.

---

getStartTime

```
public java.util.Date getStartTime()
```

Devuelve el tiempo de inicio pasado como un argumento al constructor.

Devuelve:

El tiempo de inicio pasado al constructor.

---

getDuration

```
public long getDuration()
```

Devuelve la duración pasada como argumento al constructor.

Devuelve:

La duración pasada al constructor.

D.1.6 Clase NoMoreDataEntriesException

```
org.ocap.shared.dvr
     Class NoMoreDataEntriesException

java.lang.Object
  +--java.lang.Throwable
       +--java.lang.Exception
            +--org.ocap.shared.dvr.NoMoreDataEntriesException
```

Todas las interfaces implementadas:

java.io.Serializable

---

clase pública **NoMoreDataEntriesException**

amplía java.lang.Exception

No se permiten más entradas de datos para esta petición de grabación.

Véase también:

Serialized Form

Resumen del constructor
NoMoreDataEntriesException() Construye una NoMoreDataEntriesException sin mensaje de información detallada

Métodos heredados de la clase java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Métodos heredados de la clase java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Información detallada del constructor

NoMoreDataEntriesException

public **NoMoreDataEntriesException()**

Construye una NoMoreDataEntriesException sin mensaje de información detallada

D.1.7 Interfaz ParentRecordingRequest

org.ocap.shared.dvr
Interface ParentRecordingRequest

Todas las superinterfaces:

RecordingRequest

---

interfaz pública **ParentRecordingRequest**

amplía RecordingRequest

Esta interfaz representa información correspondiente a una petición de grabación padre. La petición de grabación representada mediante esta interfaz puede tener una o más peticiones de grabación hijas.

Una petición de grabación padre puede estar en el estado de no resuelta, parcialmente resuelta, completamente resuelta o en el estado cancelado.

Una petición de grabación estará en el estado no resuelto si la implementación no tiene suficiente información para procesarla. Una petición de grabación en este estado puede pasar al estado parcialmente resuelto, completamente resuelto o al estado de fallo.

Una petición de grabación estará en el estado parcialmente resuelto si la implementación tiene información suficiente para programar alguna, aunque no todas, de las peticiones de grabación hijas de dicha petición de grabación. Ese sería el caso de una petición de grabación para una serie en la que algunos de sus episodios están programados. Una petición de grabación está en el estado completamente resuelto si se conocen y están programadas todas las grabaciones hijas.

Resumen del campo	
static int	**CANCELLED_STATE** Una petición de grabación está en estado cancelada si una aplicación ha llamado con éxito al método de cancelación (cancel) para dicha petición de grabación, pero no se han suprimido todas las peticiones de grabación hijas.
static int	**COMPLETELY_RESOLVED_STATE** Se han programado todas las grabaciones hijas correspondientes a esta petición de grabación.
static int	**PARTIALLY_RESOLVED_STATE** Se han programado algunas de las grabaciones correspondientes a esta petición de grabación.
static int	**UNRESOLVED_STATE** La implementación no tiene información suficiente para procesar esta petición de grabación.

Resumen del método	
void	**cancel**() Cancela la petición de grabación padre.
RecordingList	**getKnownChildren**() Obtiene todas las grabaciones correspondientes a esta petición de grabación padre.

Métodos heredados de la interfaz org.ocap.shared.dvr.RecordingRequest
addAppData, delete, getAppData, getAppID, getId, getKeys, getParent, getRecordingSpec, getRoot, getState, isRoot, removeAppData, reschedule, setRecordingProperties

Información detallada del campo

UNRESOLVED_STATE

```
public static final int UNRESOLVED_STATE
```

La implementación no tiene información suficiente para procesar esta petición de grabación.

Véase también:

Constant Field Values

COMPLETELY_RESOLVED_STATE

```
public static final int COMPLETELY_RESOLVED_STATE
```

Se han programado todas las grabaciones hijas correspondientes a esta petición de grabación. Una petición de grabación está en el estado completamente resuelta si la implementación tiene información suficiente para programar todas las grabaciones correspondientes a dicha petición de grabación. Una petición de grabación en el estado completamente resuelta tendría una o más grabaciones hijas.

Véase también:

Constant Field Values

PARTIALLY_RESOLVED_STATE

```
public static final int PARTIALLY_RESOLVED_STATE
```

Se han programado algunas de las grabaciones correspondientes a esta petición de grabación. Una petición de grabación está en el estado parcialmente resuelta si la implementación tiene información suficiente para programar alguna, pero no todas, de las grabaciones correspondientes a la petición de grabación. Una petición de grabación en el estado parcialmente resuelta puede tener cero, una o más grabaciones hijas.

Véase también:

Constant Field Values

CANCELLED_STATE

```
public static final int CANCELLED_STATE
```

Una petición de grabación está en estado cancelada si una aplicación ha llamado con éxito al método cancelación (cancel) para dicha petición de grabación, pero no se han suprimido todas las peticiones de grabación hijas. Una petición de grabación en este estado será suprimida por la implementación cuando hayan sido suprimidas todas las peticiones de grabación hijas.

Véase también:

Constant Field Values

Información detallada del método

cancel

```
public void cancel()
            throws java.lang.IllegalStateException,
                   AccessDeniedException
```

Cancela la petición de grabación padre. La implementación también cancelará todas las peticiones de grabación hijas llamando a sus respectivos métodos cancel(). No se programará ninguna otra grabación hija para esta petición de grabación o para alguna de sus grabaciones hijas. La petición de grabación se suprimirá de la base de datos una vez que todas las peticiones de grabación hijas se han suprimido. La cancelación de una petición de grabación padre no suprime ninguna de las grabaciones hijas que no pueden ser canceladas (es decir, si una petición de grabación hija no está en el estado pendiente). Cuando este

método finaliza con éxito, la petición de grabación se suprime de la base de datos o pasa al estado CANCELLED_STATE.

Genera:

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("cancel",..) o RecordingPermission("*",..)

`AccessDeniedException` – si no se permite que la aplicación llamante realice esta operación mediante atributos de seguridad específicos de RecordingRequest.

`java.lang.IllegalStateException` – si la petición de grabación no está en el estado UNRESOLVED_STATE, PARTIALLY_RESOLVED_STATE o COMPLETELY_RESOLVED_STATE.

---

getKnownChildren

```
public RecordingList getKnownChildren()
                              throws java.lang.IllegalStateException
```

Obtiene todas las grabaciones hijas correspondientes a esta petición de grabación padre. Para una petición de grabación que se encuentre en el estado completamente resuelto este método devuelve todos los hijos que todavía se mantienen en la base de datos del gestor de grabación (es decir, los hijos eliminados de la base de datos mediante llamadas a los métodos delete() o cancel() no estarán incluidos en la lista de grabaciones hijas). Para una petición de grabación que se encuentre en el estado parcialmente resuelta este método sólo devuelve hijos actualmente conocidos de series.

Devuelve:

La lista de grabaciones hijas correspondientes a esta grabación; el carácter nulo si no existen peticiones de grabación hijas en la base de datos RecordingManager.

Genera:

`java.lang.IllegalStateException` – si la petición de grabación no está en el estado PARTIALLY_RESOLVED_STATE o COMPLETELY_RESOLVED_STATE.

D.1.8 Interfaz RecordedService

```
org.ocap.shared.dvr
     Interface RecordedService
```

Todas las superinterfaces:

javax.tv.service.Service

---

interfaz pública **RecordedService**

amplía javax.tv.service.Service

Esta interfaz representa la parte grabada de un servicio que está siendo grabado o que fue grabado durante un cierto periodo de tiempo. Tan pronto como comienza la grabación, se crea un servicio grabado. Si la petición de grabación falla por algún motivo, normalmente se detiene la grabación y el servicio grabado queda disponible incluyendo la mayor parte del servicio grabado.

Un RecordedService tiene un atributo tiempo de medios en el que la reproducción comienza cuando el servicio grabado se selecciona en un contexto de servicio (ServiceContext). Dicho tiempo de medios es persistente y aplicable a todas las selecciones de servicio futuras por cualquier aplicación hasta que se modifique el tiempo de medios mediante una llamada a setMediaTime.

Obsérvese el comportamiento específico de la subinterfaz para métodos definidos mediante la superinterfaz javax.tv.service.Service:

– El método hasMultipleInstances() siempre devolverá el valor falso.

– El método getServiceType() siempre devolverá RecordedServiceType.RECORDED_SERVICE.

– El método getLocator() devolverá un localizador correspondiente al servicio grabado, diferente del localizador del servicio originario. Cuando este localizador pasa al SIManager.getService(..) devuelve el RecordedService.

– El método getName() devolverá una cadena legible por personas.

– RecordedServices no estará incluido en las listas de servicio devueltas por el método SIManager.filterServices(..).

Resumen del método	
`void`	**`delete`**`()` Suprime el servicio grabado.
`javax.media.Time`	**`getFirstMediaTime`**`()` Obtiene el tiempo de medios JMF al inicio del servicio grabado.
`javax.media.MediaLocator`	**`getMediaLocator`**`()` Devuelve el MediaLocator correspondiente al RecordedService.
`javax.media.Time`	**`getMediaTime`**`()` Obtiene el tiempo de medios JMF que se fijó utilizando el método setMediaTime(..)
`long`	**`getRecordedDuration`**`()` Obtiene la duración real del contenido grabado.
`RecordingRequest`	**`getRecordingRequest`**`()` Obtienen el objeto `RecordingRequest` correspondiente al RecordedService.
`java.util.Date`	**`getRecordingStartTime`**`()` Obtiene el tiempo en que se inició la grabación.
`void`	**`setMediaTime`**`(javax.media.Time mediaTime)` Fija el tiempo de medios JMF para la ubicación desde donde comienza la reproducción cuando este servicio grabado se selecciona en un ServiceContext.

Métodos heredados de la interfaz javax.tv.service.Service
`equals, getLocator, getName, getServiceType, hashCode, hasMultipleInstances, retrieveDetails`

Información detallada del método

getRecordingRequest

```
public RecordingRequest getRecordingRequest()
```

Obtienen el objeto `RecordingRequest` correspondiente al RecordedService.

Devuelve:

El objeto `RecordingRequest` para el servicio.

---

getRecordedDuration

```
public long getRecordedDuration()
```

Obtiene la duración real del contenido grabado. Para las grabaciones en curso devuelve la duración de la parte finalizada de la grabación.

Devuelve:

La duración de la grabación en milisegundos.

---

getMediaLocator

```
public javax.media.MediaLocator getMediaLocator()
```

Devuelve el MediaLocator correspondiente al RecordedService.

Devuelve:

MediaLocator del RecordedService.

---

setMediaTime

```
public void setMediaTime(javax.media.Time mediaTime)
                  throws AccessDeniedException
```

Fija el tiempo de medios JMF para la ubicación desde donde comienza la reproducción cuando este servicio grabado se selecciona en un ServiceContext.

Si se fija una instancia de tiempo con un valor de 0 nanosegundos o un valor negativo, o bien si este método no ha sido llamado para este servicio grabado, la reproducción comienza en el punto de inicio del contenido grabado. Si la instancia de fijación de tiempo (Timeset) es infinito positivo o un valor superior a la duración del contenido grabado, la reproducción comienza en el punto en directo si la grabación está en curso para el servicio grabado. En caso de que la grabación no esté en curso, la reproducción alcanzará inmediatamente el fin-de-medios. NOTA – El tiempo de medios fijado será aplicable a todas las selecciones de servicio futuras realizadas por todas las aplicaciones.

Parámetros:

`mediaTime` – tiempo de medios correspondiente al punto desde el que comienza la reproducción cuando se selecciona el servicio.

Genera:

`AccessDeniedException` – si no se permite a la aplicación llamante realizar esta operación mediante atributos de seguridad específicos de RecordingRequest correspondientes a la RecordingRequest asociada con este servicio grabado.

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("modify",..) o RecordingPermission("*",..)

---

getMediaTime

```
public javax.media.Time getMediaTime()
```

Obtiene el tiempo de medios JMF que se fijó utilizando el método setMediaTime(..)

Devuelve:

El valor del tiempo de medios JMF que se fijó utilizando el método setMediaTime(..) si dicho método había sido anteriormente llamado en este RecordedService. Si no se había llamado anteriormente al método setMediaTime, este método debería devolver el tiempo de medios JMF correspondiente al comienzo del RecordedService.

---

getRecordingStartTime

```
public java.util.Date getRecordingStartTime()
```

O tiempo en que se inició la grabación.

Devuelve:

El tiempo en que la implementación inició la grabación.

---

delete

```
public void delete()
            throws AccessDeniedException
```

Suprime el servicio grabado. El método elimina el servicio grabado y todos los flujos elementales grabados (por ejemplo, ficheros y apuntes en directorios) asociados con el RecordedService. La correspondiente petición de grabación pasará al estado DELETED_STATE.

Si la petición de grabación se encuentra en el estado IN_PROGRESS la implementación detendrá la grabación antes de suprimir el servicio grabado. Si el servicio RecordedService estaba siendo presentado cuando fue suprimido, se enviará un `PresentationTerminatedEvent` con el motivo SERVICE_VANISHED.

Genera:

`AccessDeniedException` – si no se permite que la aplicación llamante realice esta operación mediante atributos de seguridad específicos de RecordingRequest.

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("delete",..) o RecordingPermission("*",..)

---

getFirstMediaTime

```
public javax.media.Time getFirstMediaTime()
```

Obtiene el tiempo de medios JMF al inicio del servicio grabado.

Devuelve:

Un tiempo de medios

D.1.9 Clase RecordedServiceType

```
org.ocap.shared.dvr
     Class RecordedServiceType

java.lang.Object
  +--javax.tv.service.ServiceType
       +--org.ocap.shared.dvr.RecordedServiceType
```

---

clase pública **RecordedServiceType**

amplía javax.tv.service.ServiceType

Esta clase representa el valor del tipo de servicio (ServiceType) para un servicio grabado (RecordedService).

Resumen del campo	
static javax.tv.service.ServiceType	**RECORDED_SERVICE_TYPE** Tipo de servicio (ServiceType) para un servicio grabado.

Campos heredados de la clase javax.tv.service.ServiceType
ANALOG_RADIO, ANALOG_TV, DATA_APPLICATION, DATA_BROADCAST, DIGITAL_RADIO, DIGITAL_TV, NVOD_REFERENCE, NVOD_TIME_SHIFTED, UNKNOWN

Resumen del constructor	
protected	**RecordedServiceType**(java.lang.String name) Proporciona una instancia de tipo de servicio grabado (RecordedServiceType).

Métodos heredados de la clase javax.tv.service.ServiceType
toString

Métodos heredados de la clase java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Información detallada del campo

RECORDED_SERVICE_TYPE

```
public static final javax.tv.service.ServiceType RECORDED_SERVICE_TYPE
```

ServiceType para un servicio grabado.

Información detallada del constructor

RecordedServiceType

```
protected RecordedServiceType(java.lang.String name)
```

Proporciona una instancia de tipo de servicio grabado (RecordedServiceType).

Parámetros:

`name` – el nombre de la cadena de este tipo (es decir, "RECORDED_SERVICE").

D.1.10 Clase RecordingChangedEvent

```
org.ocap.shared.dvr
     Class RecordingChangedEvent

java.lang.Object
  +--java.util.EventObject
       +--org.ocap.shared.dvr.RecordingChangedEvent
```

Todas las interfaces implementadas:

java.io.Serializable

---

clase pública **RecordingChangedEvent**

amplía java.util.EventObject

Evento utilizado para notificar a los oyentes los cambios en la lista de peticiones de grabación que mantiene el RecordingManager.

Véase también:

Serialized Form

Resumen del campo

Static int	**ENTRY_ADDED** Se ha añadido una nueva `RecordingRequest`.
Static int	**ENTRY_DELETED** Se ha suprimido una `RecordingRequest`.
Static int	**ENTRY_STATE_CHANGED** Ha cambiado el estado de una `RecordingRequest`.

Campos heredados de la clase java.util.EventObject

Source

Resumen del constructor
RecordingChangedEvent(RecordingRequest source, int newState, int oldState) Construye el evento.

Resumen del método	
int	**getChange**() Devuelve la modificación al RecordingRequest.
int	**getOldState**() Devuelve el estado anterior de RecordingRequest.
RecordingRequest	**getRecordingRequest**() Devuelve el RecordingRequest que causó el evento.
int	**getState**() Devuelve el nuevo estado de RecordingRequest.

Métodos heredados de la clase java.util.EventObject
getSource, toString

Métodos heredados de la clase java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Información detallada del campo

ENTRY_ADDED

public static final int **ENTRY_ADDED**

Se ha añadido una nueva RecordingRequest.

Véase también:

Constant Field Values

ENTRY_DELETED

public static final int **ENTRY_DELETED**

Se ha suprimido una RecordingRequest.

Véase también:

Constant Field Values

ENTRY_STATE_CHANGED

```
public static final int ENTRY_STATE_CHANGED
```

Ha cambiado el estado de una `RecordingRequest`

Véase también:

Constant Field Values

Información detallada del constructor

RecordingChangedEvent

```
public RecordingChangedEvent(RecordingRequest source,
                             int newState,
                             int oldState)
```

Construye el evento.

Parámetros:

`source` – el `RecordingRequest` que causó el evento.

`newState` – el estado en que ahora se encuentra el `RecordingRequest`.

`oldState` – el estado en que estaba el `RecordingRequest` antes de la modificación de estado.

Información detallada del método

getRecordingRequest

```
public RecordingRequest getRecordingRequest()
```

Devuelve el `RecordingRequest` que causó el evento.

Devuelve:

El `RecordingRequest` que causó el evento.

---

getChange

```
public int getChange()
```

Devuelve la modificación a la `RecordingRequest`.

Devuelve:

Si la entrada se añadió, se suprimió o se modificó.

---

getState

```
public int getState()
```

Devuelve el nuevo estado de la `RecordingRequest`.

Devuelve:

El nuevo estado.

---

getOldState

```
public int getOldState()
```

Devuelve el estado anterior de `RecordingRequest`.

Devuelve:

El estado anterior.

D.1.11 Interfaz RecordingChangedListener

```
org.ocap.shared.dvr
     Interface RecordingChangedListener
```

Todas las superinterfaces:

java.util.EventListener

---

interfaz pública **RecordingChangedListener**

amplía java.util.EventListener

Oyente que recibe cambios en la lista de grabaciones que mantiene el RecordingManager.

Resumen del método	
void	**recordingChanged**(RecordingChangedEvent e) Notifica al `RecordingChangedListener` de un evento generado por el `RecordingManager`.

Información detallada del método

recordingChanged

```
public void recordingChanged(RecordingChangedEvent e)
```

Notifica al `RecordingChangedListener` de un evento generado por el `RecordingManager`. Los eventos se generan cuando existen modificaciones en la lista de peticiones de grabación que mantiene el gestor de grabaciones.

Parámetros:

`e` – El evento generado.

D.1.12 Clase RecordingFailedException

```
org.ocap.shared.dvr
     Class RecordingFailedException

java.lang.Object
  +--java.lang.Throwable
       +--java.lang.Exception
            +--org.ocap.shared.dvr.RecordingFailedException
```

Todas las interfaces implementadas:

java.io.Serializable

---

clase pública **RecordingFailedException**

amplía java.lang.Exception

Esta excepción se devuelve cuando las aplicaciones llaman a getFailedException() en caso de una petición de grabación fallida o de una petición de grabación incompleta.

Véase también:

Serialized Form

Resumen del campo	
static int	**ACCESS_WITHDRAWN** Código de motivo: La grabación no se completó con éxito porque el acceso al servicio o algún componente del mismo fue retirado por el sistema antes de la compleción programada de la grabación.
static int	**CA_REFUSAL** Código de motivo: La grabación falló porque el sistema de CA system no la permitió.
static int	**CONTENT_NOT_FOUND** Código de motivo: La grabación falló porque el contenido requerido no se encontró en la red.
static int	**INSUFFICIENT_RESOURCES** Código de motivo: La grabación falló debido a la escasez de recursos requeridos para presentar este servicio.
static int	**OUT_OF_BANDWIDTH** Código de motivo: La grabación falló debido a la escasez de anchura de banda de IO para grabar este programa.
static int	**RESOLUTION_ERROR** Código de motivo: La petición de grabación falló debido a errores en la resolución de la petición.
static int	**RESOURCES_REMOVED** Código de motivo: La grabación no pudo completarse con éxito debido a que los recursos necesarios para presentar el servicio habían sido eliminados antes de la compleción programada de la grabación.
static int	**SERVICE_VANISHED** Código de motivo: La grabación no pudo completarse con éxito debido a que el servicio desapareció de la red antes de la compleción de la grabación.
static int	**SPACE_FULL** Código de motivo: La grabación no pudo completarse por falta de espacio de almacenamiento.
static int	**TUNED_AWAY** Código de motivo: La grabación no pudo completarse con éxito debido a que la aplicación seleccionó otro servicio en el contexto de servicio.
static int	**TUNING_FAILURE** Código de motivo: La grabación falló debido a problemas de sintonización.
static int	**USER_STOP** Código de motivo: La aplicación terminó la grabación utilizando el método LeafRecordingRequest.stop o haciendo una llamada a la parada (stop) en el contexto del servicio (si la especificación de grabación es una instancia de ServiceContextRecordingSpec).

Resumen del constructor
RecordingFailedException() Construye una RecordingFailedException sin mensaje de información detallada

Resumen del método	
int	**getReason**() Informa del motivo por el que la petición de grabación falló y no se completó con éxito.

Métodos heredados de la clase java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, printStackTrace, toString

Métodos heredados de la clase java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Información detallada del campo

CA_REFUSAL

public static final int **CA_REFUSAL**

Código de motivo: La grabación falló porque el sistema de acceso condicional no lo permitió.

Véase también:

Constant Field Values

CONTENT_NOT_FOUND

public static final int **CONTENT_NOT_FOUND**

Código de motivo: La grabación falló porque el contenido requerido no se encontró en la red.

Véase también:

Constant Field Values

TUNING_FAILURE

public static final int **TUNING_FAILURE**

Código de motivo: La grabación falló debido a problemas de sintonización.

Véase también:

Constant Field Values

INSUFFICIENT_RESOURCES

```
public static final int INSUFFICIENT_RESOURCES
```

Código de motivo: La grabación falló debido a escasez de recursos necesarios para presentar este servicio.

Véase también:

Constant Field Values

ACCESS_WITHDRAWN

```
public static final int ACCESS_WITHDRAWN
```

Código de motivo: La grabación no se completó con éxito porque el acceso al servicio o algún componente del mismo fue retirado por el sistema antes de la compleción programada de la grabación.

Véase también:

Constant Field Values

RESOURCES_REMOVED

```
public static final int RESOURCES_REMOVED
```

Código de motivo: La grabación no pudo completarse con éxito debido a que los recursos necesarios para presentar el servicio habían sido eliminados antes de la compleción programada de la grabación.

Véase también:

Constant Field Values

SERVICE_VANISHED

```
public static final int SERVICE_VANISHED
```

Código de motivo: La grabación no pudo completarse con éxito debido a que el servicio desapareció de la red antes de la compleción de la grabación.

Véase también:

Constant Field Values

TUNED_AWAY

```
public static final int TUNED_AWAY
```

Código de motivo: La grabación no pudo completarse con éxito debido a que la aplicación seleccionó otro servicio en el contexto de servicio. Sólo es aplicable si la especificación de grabación para la petición de grabación es una instancia de ServiceContextRecordingSpec.

Véase también:

Constant Field Values

USER_STOP

```
public static final int USER_STOP
```

Código de motivo: La aplicación terminó la grabación utilizando el método LeafRecordingRequest.stop o haciendo una llamada a la parada (stop) en el contexto del servicio (si la especificación de la grabación es una instancia de ServiceContextRecordingSpec).

Véase también:

Constant Field Values

SPACE_FULL

```
public static final int SPACE_FULL
```

Código de motivo: La grabación no pudo completarse por falta de espacio de almacenamiento.

Véase también:

Constant Field Values

OUT_OF_BANDWIDTH

```
public static final int OUT_OF_BANDWIDTH
```

Código de motivo: La grabación falló debido a escasez de la anchura de banda de IO para grabar el programa en cuestión.

Véase también:

Constant Field Values

RESOLUTION_ERROR

```
public static final int RESOLUTION_ERROR
```

Código de motivo: La petición de grabación falló debido a errores en la resolución de la petición.

Véase también:

Constant Field Values

Información detallada del constructor

RecordingFailedException

```
public RecordingFailedException()
```

Construye una RecordingFailedException sin mensaje de información detallada

Información detallada del método

getReason

```
public int getReason()
```

Informa del código de motivo por el que la petición de grabación falló y no se completó con éxito.

Devuelve:

El código de motivo por el que la petición de grabación falló y no se completó con éxito.

D.1.13 Clase RecordingManager

```
org.ocap.shared.dvr
     Class RecordingManager

java.lang.Object
  +--org.ocap.shared.dvr.RecordingManager
```

---

clase abstracta pública **RecordingManager**

amplía java.lang.Object

RecordingManager representa la entidad que realiza grabaciones.

Resumen del constructor	
protected	**RecordingManager**() Constructor para instancias de esta clase.

Resumen del método	
abstract void	**addRecordingChangedListener**(RecordingChangedListener rcl) Añade un oyente de eventos cuando se producen cambios en el estado de peticiones de grabación.
abstract RecordingList	**getEntries**() Obtiene la lista de entradas que mantiene el RecordingManager.
abstract RecordingList	**getEntries**(RecordingListFilter filter) Obtiene la lista de peticiones de grabación que concuerdan con el filtro especificado.
static RecordingManager	**getInstance**() Obtiene la instancia singleton del RecordingManager.
abstract RecordingRequest	**getRecordingRequest**(int id) Detecta una petición de grabación del identificador.
abstract RecordingRequest	**record**(RecordingSpec source) Graba el flujo o los flujos de acuerdo con el parámetro fuente.
abstract void	**removeRecordingChangedListener** (RecordingChangedListener rcl) Elimina un oyente de evento registrado en el caso de cambios de estado de las peticiones de grabación.

Métodos heredados de la clase java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Información detallada del constructor

RecordingManager

```
protected RecordingManager()
```

Constructor para instancias de esta clase. Este constructor permite la utilización de implementaciones y especificaciones que amplíen esta especificación. Las aplicaciones no definirán subclases de esta clase. No se requiere que las implementaciones se comporten correctamente si se utiliza cualquiera de las subclases definidas en función de la aplicación.

Información detallada del método

getEntries

```
public abstract RecordingList getEntries()
```

Obtiene la lista de entradas que mantiene el RecordingManager. Esta lista incluye peticiones de grabación padres y peticiones de grabación hojas. En el caso de aplicaciones con RecordingPermission("read", "own"), sólo se devuelven RecordingRequests de las que la aplicación llamante tenga visibilidad, tal como se define mediante cualquiera de los atributos de seguridad específicos de RecordingRequest. en el caso de aplicaciones con RecordingPermission("read", "*"), se devuelven todas las RecordingRequests.

Devuelve:

Una instancia de RecordingList.

Genera:

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("read",..) o RecordingPermission("*",..)

---

getEntries

```
public abstract RecordingList getEntries(RecordingListFilter filter)
```

Obtiene la lista de peticiones de grabación que concuerdan con el filtro especificado. En el caso de aplicaciones con RecordingPermission("read", "own"), sólo se devuelven RecordingRequests de las que la aplicación llamante tenga visibilidad, tal como se define mediante cualquiera de los atributos de seguridad específicos de RecordingRequest. Para aplicaciones con RecordingPermission("read", "*"), se devolverán todas las RecordingRequests de todas las modificaciones que concuerden con el filtro especificado.

Parámetros:

`filter` – el filtro a utilizar en el conjunto total de peticiones de grabación.

Devuelve:

Una instancia de RecordingList.

Genera:

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("read",..) o RecordingPermission("*",..)

---

addRecordingChangedListener

```
public abstract void addRecordingChangedListener(RecordingChangedListener rcl)
```

Añade un oyente de eventos cuando se producen cambios en el estado de peticiones de grabación. En el caso de aplicaciones con RecordingPermission("read", "own"), el parámetro oyente sólo será informado de modificaciones que afecten a RecordingRequests de las que la aplicación llamante tenga visibilidad, tal como se define mediante cualquiera de los atributos de seguridad específicos de RecordingRequest. Para aplicaciones con RecordingPermission("read", "*"), el parámetro oyente será informado de todas las modificaciones.

Parámetros:

`rcl` – el oyente que debe ser registrado.

Genera:

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("read",..) o RecordingPermission("*",..)

---

removeRecordingChangedListener

```
public abstract void
removeRecordingChangedListener(RecordingChangedListener rcl)
```

Elimina un oyente de evento registrado en el caso de cambios de estado de las peticiones de grabación. Si el oyente especificado no está registrado, este método no tiene efecto alguno.

Parámetros:

`rcl` – el oyente que debe ser eliminado.

---

record

```
public abstract RecordingRequest record(RecordingSpec source)
                                 throws java.lang.IllegalArgumentException,
                                        AccessDeniedException
```

Graba el flujo o los flujos de acuerdo con el parámetro fuente. La subclase concreta de RecordingSpec puede definir una semántica adicional que se aplique cuando se utilizan instancias de dicha subclase.

Parámetros:

`source` – especificación del flujo o flujos que han de ser grabados y cómo han de serlo.

Devuelve:

Una instancia de RecordingRequest que representa la grabación añadida.

Genera:

`java.lang.IllegalArgumentException` – si la fuente es una clase definida de aplicación o según se define en la subclase concreta de RecordingSpec para instancias de dicha clase.

AccessDeniedException – si no se permite que la aplicación llamante realice esta operación mediante atributos de seguridad específicos de RecordingRequest.

java.lang.SecurityException – si la aplicación llamante no tiene RecordingPermission("create",..) o RecordingPermission("*",..)

---

getInstance

```
public static RecordingManager getInstance()
```

Obtiene la instancia singleton del RecordingManager.

Devuelve:

Una instancia del RecordingManager.

---

getRecordingRequest

```
public abstract RecordingRequest getRecordingRequest(int id)
                                          throws java.lang.IllegalArgumentException
```

Detecta una petición de grabación del identificador. Las implementaciones de este método deberían ser optimizadas considerando que probablemente el número de peticiones de grabación sea muy alto. En el caso de aplicaciones con RecordingPermission("read", "own"), sólo se devuelven RecordingRequests de las que la aplicación llamante tenga visibilidad, tal como se define mediante cualquiera de los atributos de seguridad específicos de RecordingRequest.

Parámetros:

id – un identificador tal como es devuelto por RecordingRequest.getId

Devuelve:

La RecordingRequest correspondiente.

Genera:

java.lang.IllegalArgumentException – si no existe una petición de grabación correspondiente a este identificador o si la petición de grabación no es visible tal como definen los atrib is not visible as tos de seguridad específicos de RecordingRequest.

java.lang.SecurityException – si la aplicación llamante no tiene RecordingPermission("read",..) o RecordingPermission("*",..)

Véase también:

RecordingRequest.getId()

D.1.14 Clase RecordingPermission

```
org.ocap.shared.dvr
     Class RecordingPermission

java.lang.Object
  +--java.security.Permission
       +--org.ocap.shared.dvr.RecordingPermission
```

Todas las interfaces implementadas:

java.security.Guard, java.io.Serializable

---

clase final pública **RecordingPermission**

amplía java.security.Permission

Controla el acceso que puede tener una aplicación a las características de grabación. El nombre puede ser uno de los que se muestran en la lista siguiente:

– "create" (*creación*) – programa una RecordingRequest (petición de grabación)
– "read" (*lectura*) – obtiene la lista de RecordingRequests
– "modify" (*modificación*) – modifica propiedades o datos específicos de la aplicación para una RecordingRequest
– "delete" (*supresión*) – suprime una RecordingRequest incluyendo el contenido grabado
– "cancel" (*cancelación*) – cancela una RecordingRequest pendiente
– "*" – todo lo anterior.

La acción puede ser "own" y "*". La acción "own" está previsto que pueda ser utilizada por aplicaciones normales. La acción "*" sólo está previsto que sea utilizada por aplicaciones privilegiadas y permite que la operación definida por el nombre sea aplicable a todas las RecordingRequests con independencia de las restricciones asociadas a la RecordingRequest que pueda tener cada aplicación.

La concesión de este permiso incluirá la concesión de acceso a cualquier dispositivo de almacenamiento que sea necesario para las operaciones especificadas en el parámetro nombre. No son necesarios permisos adicionales de nivel inferior (por ejemplo, FilePermission) ulteriores.

Véase también:

Serialized Form

Resumen del constructor
`RecordingPermission``(java.lang.String name, java.lang.String action)` Crea un nuevo RecordingPermission con el nombre y la acción especificados.

Resumen del método	
`boolean`	**`equals`**`(java.lang.Object obj)` Verifica la igualdad de dos objetos RecordingPermission (*permiso de grabación*).
`java.lang.String`	**`getActions`**`()` Devuelve las acciones tal como se pasan al constructor.
`int`	**`hashCode`**`()` Devuelve valor del código resumen (*hash*) de este objeto.
`boolean`	**`implies`**`(java.security.Permission p)` Verifica si este RecordingPermission "implica" el permiso (Permission) especificado.

Métodos heredados de la clase java.security.Permission
`checkGuard, getName, newPermissionCollection, toString`

Métodos heredados de la clase java.lang.Object

`clone, finalize, getClass, notify, notifyAll, wait, wait, wait`

Información detallada del constructor

RecordingPermission

```
public RecordingPermission(java.lang.String name,
                           java.lang.String action)
```

Crea un nuevo RecordingPermission con el nombre y la acción especificados

Parámetros:

`name` – "create", "read", "modify", "delete", "cancel" o "*"

`action` – "own" o "*"

Información detallada del método

implies

```
public boolean implies(java.security.Permission p)
```

Verifica si este RecordingPermission "implica" el permiso (Permission) especificado.

Parámetros:

`p` – el permiso que debe verificarse

Devuelve:

"verdadero" si el objeto implica el permiso especificado, "falso" si no lo implica.

---

equals

```
public boolean equals(java.lang.Object obj)
```

Verifica la igualdad de dos objetos RecordingPermission (*permiso de grabación*)

Parámetros:

`obj` – el objeto del que verificar la igualdad con este objeto.

Devuelve:

"verdadero" si obj es un RecordingPermission con los mismos nombre y acción que este objeto RecordingPermission.

---

hashCode

```
public int hashCode()
```

Devuelve valor del código resumen (*hash*) de este objeto.

Devuelve:

Un valor de código resumen (*hash*) de este objeto.

getActions

```
public java.lang.String getActions()
```

Devuelve las acciones tal como se pasan al constructor.

Devuelve:

Las acciones en forma de cadena (*String*)

D.1.15 Clase RecordingProperties

```
org.ocap.shared.dvr
     Class RecordingProperties

java.lang.Object
  +--org.ocap.shared.dvr.RecordingProperties
```

clase abstracta pública **RecordingProperties**

amplía java.lang.Object

Clase base para especificar propiedades que definen cómo se realiza una grabación.

Resumen del constructor
RecordingProperties(long expirationPeriod) Constructor

Resumen del método	
long	**getExpirationPeriod**() Devuelve el valor del periodo de expiración.

Métodos heredados de la clase java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Información detallada del constructor

RecordingProperties

```
public RecordingProperties(long expirationPeriod)
```

Constructor

Parámetros:

expirationPeriod – el periodo en segundos, después del inicio de una grabación, tras el que se considera que han expirado las peticiones de grabación hojas con esta propiedad de grabación.

Información detallada del método

getExpirationPeriod

```
public long getExpirationPeriod()
```

Devuelve el valor del periodo de expiración.

Devuelve:

El periodo de expiración tal como se pasa al constructor.

D.1.16 Interfaz RecordingRequest

```
org.ocap.shared.dvr
     Interface RecordingRequest
```

Todas las subinterfaces conocidas:

LeafRecordingRequest, ParentRecordingRequest

Interfaz pública **RecordingRequest**

Esta interfaz representa información correspondiente a una petición de grabación. La petición de grabación que representa esta interfaz puede corresponder a una única petición de grabación o a series de peticiones de grabación. Las peticiones de grabación son de naturaleza jerárquica. Las implementaciones pueden resolver una petición de grabación en una única petición de grabación o en una serie de peticiones de grabación, cada una de las cuales puede, a su vez, resolverse mediante una única petición de grabación o una serie de peticiones de grabación. Por ejemplo, una petición de grabación para "Sex and the City" puede resolverse en múltiples peticiones de grabación, cada una para una temporada específica. Cada una de dichas peticiones de grabación puede resolverse en múltiples peticiones de grabación, una para cada episodio. La implementación crea una petición de grabación en respuesta al método RecordingManager.record(RecordingSpec). La implementación también crea peticiones de grabación cuando se resuelve una petición de grabación.

Una petición de grabación puede ser petición de grabación padre o petición de grabación hoja. Los estados de una petición de grabación se definen en `ParentRecordingRequest` y `LeafRecordingRequest`. Una petición de grabación puede estar en cualquiera de los estados correspondientes a una petición de grabación hoja o a una petición de grabación padre.

Resumen del método	
void	**addAppData**(java.lang.String key, java.io.Serializable data) Añade datos privados específicos de la aplicación.
void	**delete**() Suprime la petición de grabación de la base de datos.
java.io.Serializable	**getAppData**(java.lang.String key) Obtiene datos de aplicación correspondientes a la clave especificada.
org.dvb.application.AppID	**getAppID**() Obtiene el identificador de aplicación de la aplicación que posee esta petición de grabación.
int	**getId**() Devuelve un identificador para esta petición de grabación.
java.lang.String[]	**getKeys**() Obtiene todos los datos específicos de la aplicación asociados a esta petición de grabación.
RecordingRequest	**getParent**() Obtiene la petición de grabación padre correspondiente a esta petición de grabación.
RecordingSpec	**getRecordingSpec**() Devuelve la the RecordingSpec correspondiente a esta petición de grabación.
RecordingRequest	**getRoot**() Obtiene la petición de grabación raíz correspondiente a esta petición de grabación.
int	**getState**() Devuelve el estado de la petición de grabación.
boolean	**isRoot**() Verifica si la petición de grabación era una petición de grabación raíz generada cuando la aplicación hizo una llamada a RecordingManager.record(..)
void	**removeAppData**(java.lang.String key) Elimina datos privados específicos de la aplicación correspondientes a la clave especificada.
void	**reschedule**(RecordingSpec newRecordingSpec) Modifica la información detallada de una petición de grabación.
Void	**setRecordingProperties**(RecordingProperties properties) Modifica las RecordingProperties correspondientes a la RecordingSpec de esta petición de grabación.

Información detallada del método

getState

```
public int getState()
```

Devuelve el estado de la petición de grabación.

Devuelve:

Estado de la petición de grabación.

---

isRoot

```
public boolean isRoot()
```

Verifica si la petición de grabación era una petición de grabación raíz generada cuando la aplicación hizo una llamada al método RecordingManager.record(..). La implementación debería crear una petición de grabación raíz correspondiente a cada llamada exitosa realizada al método de grabación.

Devuelve:

Verdadero, si la petición de grabación es una petición de grabación raíz, falso, si la petición de grabación se generó durante el proceso de resolución de otra petición de grabación.

---

getRoot

```
public RecordingRequest getRoot()
```

Obtiene la petición de grabación raíz correspondiente a esta petición de grabación. Una petición de grabación es la petición de grabación que se ha devuelto cuando la aplicación ha hecho una llamada al método RecordingManager.record(..).

Si la petición de grabación actual es una petición de grabación raíz, se devuelve la actual petición de grabación.

Devuelve:

La petición de grabación raíz de esta petición de grabación, nada si la aplicación no tiene permiso de acceso de lectura para la petición de grabación raíz.

---

getParent

```
public RecordingRequest getParent()
```

Obtiene la petición de grabación padre correspondiente a esta petición de grabación.

Devuelve:

La petición de grabación padre de esta petición de grabación, nada si la aplicación no tiene permiso de acceso de lectura para la petición de grabación padre, o si esta petición de grabación es la petición de grabación raíz.

---

getRecordingSpec

```
public RecordingSpec getRecordingSpec()
```

Devuelve la RecordingSpec correspondiente a esta petición de grabación. Podrá ser o bien la fuente, tal como se especifica en la llamada al método record(..) que ha motivado la creación de esta petición de grabación, o la RecordingSpec generada por el sistema durante la resolución de la RecordingSpec especificada por la aplicación original. Cualquier modificación de RecordingSpec debida a ulteriores llamadas a los métodos SetRecordingProperties sobre esta instancia se reflejará en la RecordingSpec devuelta.

Cuando la implementación genera una petición de grabación al tiempo que resuelve otra petición de grabación, se crea una nueva instancia de RecordingSpec con una copia idéntica de las RecordingProperties de la petición de grabación padre.

Devuelve:

Una RecordingSpec que contiene información sobre esta petición de grabación.

---

setRecordingProperties

```
public void setRecordingProperties(RecordingProperties properties)
                            throws java.lang.IllegalStateException,
                                   AccessDeniedException
```

Modifica las RecordingProperties correspondientes a la RecordingSpec de esta petición de grabación. Las aplicaciones pueden modificar cualquiera de las propiedades asociadas a una petición de grabación haciendo una llamada a este método. La modificación de las propiedades puede dar lugar a cambios de estado de esta petición de grabación. La modificación de las propiedades de una petición de grabación padre no modificará automáticamente las propiedades de sus peticiones de grabación hijas que ya hayan sido creadas. Cualquier petición de grabación hija creada después de la invocación de este método, heredará los nuevos valores para las propiedades.

Parámetros:

`properties` – las nuevas propiedades de grabación a establecer.

Genera:

`java.lang.IllegalStateException` – si la modificación de uno de los parámetros que ha sido modificado en las nuevas propiedades de grabación no es legal en el estado actual de la petición de grabación.

`AccessDeniedException` – si no se permite que la aplicación llamante realice esta operación mediante atributos de seguridad específicos de RecordingRequest.

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("modify",..) or RecordingPermission("*",..)

---

delete

```
public void delete()
            throws AccessDeniedException
```

Suprime la petición de grabación de la base de datos. Este método elimina la petición de grabación, todas sus peticiones de grabación descendientes, los correspondientes objetos `RecordedService` y todos los flujos elementales grabados (por ejemplo, ficheros y apuntes del directorio) asociados con el RecordedService. Si una aplicación realiza una llamada a un método cualquiera con preferencias ya obsoletas de objetos eliminados, la implementación generará una IllegalStateException.

Si la petición de grabación está en el estado the IN_PROGRESS, la implementación detendrá la grabación antes de suprimir la petición de grabación. Si se estaba presentado un RecordedService cuando fue suprimida, se enviará una `PresentationTerminatedEvent` con el motivo SERVICE_VANISHED.

Genera:

`AccessDeniedException` – si no se permite que la aplicación llamante realice esta operación mediante atributos de seguridad específicos de RecordingRequest.

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("delete",..) o RecordingPermission("*",..)

---

addAppData

```
public void addAppData(java.lang.String key,
                       java.io.Serializable data)
                throws NoMoreDataEntriesException,
                       AccessDeniedException
```

Añade datos privados específicos de la aplicación Si la clave ya está siendo utilizada, se sobre escriben los datos correspondientes.

Parámetros:

`key` – el ID al amparo del cual se añaden los datos

`data` – los datos que se añaden

Genera:

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("modify",..) or RecordingPermission("*",..)

`java.lang.IllegalArgumentException` – si el tamaño de los datos es mayor que el tamaño que soporta la implementación dentro de las limitaciones de la especificación de grabación GEM.

`NoMoreDataEntriesException` – si el almacenamientos de estos datos hace que se superen el número de apuntes que soporta la implementación dentro de las limitaciones de la especificación de grabación GEM.

`AccessDeniedException` – si no se permite que la aplicación llamante realice esta operación mediante atributos de seguridad específicos de RecordingRequest.

---

getAppID

```
public org.dvb.application.AppID getAppID()
```

Obtiene el identificador de aplicación de la aplicación que posee dicha petición de grabación. El propietario de una petición de grabación raíz es la aplicación que hizo la llamada al método RecordingManager.record(..). El propietario de una petición de grabación no raíz es el propietario de la raíz de la petición de grabación.

Devuelve:

Identificador de aplicación de la aplicación propietaria.

---

getKeys

```
public java.lang.String[] getKeys()
```

Obtiene todos los datos específicos de la aplicación asociados a esta petición de grabación.

Devuelve:

Todas las claves correspondientes el RecordingRequest; nada si no hay datos de aplicación.

---

getAppData

```
public java.io.Serializable getAppData(java.lang.String key)
```

Obtiene datos de aplicación correspondientes a la clave especificada.

Parámetros:

`key` – la clave con la que se devuelve cualquier dato.

Devuelve:

Los datos de aplicación correspondientes a la clave especificada; nada si no hay datos correspondientes a la clave especificada.

---

removeAppData

```
public void removeAppData(java.lang.String key)
                   throws AccessDeniedException
```

Elimina datos privados específicos de la aplicación correspondientes a la clave especificada. Este método deja de funcionar de forma silenciosa en caso de que no hubiera datos correspondientes a la clave especificada.

Parámetros:

`key` – la clave con la que se eliminan los datos.

Genera:

`AccessDeniedException` – si no se permite que la aplicación llamante realice esta operación mediante atributos de seguridad específicos de RecordingRequest.

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("modify",..) or RecordingPermission("*",..)

---

reschedule

```
public void reschedule(RecordingSpec newRecordingSpec)
                throws AccessDeniedException
```

Modifica la información detallada de una petición de grabación. La petición de grabación será reevaluada en base a las nuevas RecordingSpec. La reprogramación de una petición de grabación raíz puede dar lugar a cambios de estado de la petición de grabación raíz o de sus peticiones de grabación hijas. La reprogramación de una petición de grabación raíz también puede provocar la reprogramación de una o más nuevas peticiones de grabación hijas, o la supresión de una o más peticiones de grabación hijas pendientes.

NOTA – Si la petición de grabación o una de sus peticiones de grabación hijas se encuentre en el estado IN_PROGRESS_STATE o IN_PROGRESS_INSUFFICIENT_SPACE_STATE se ignorará cualquier cambio del tiempo de inicio. En este caso, se aplican todos los restantes parámetros válidos. Si el nuevo valor de un parámetro no es válido (por ejemplo, el tiempo de inicio y la duración pertenecen al pasado), la implementación ignorará dicho parámetro. Las peticiones de grabación que estén en curso continuarán sin ser interrumpidas si la nueva especificación de grabación no requiere que la grabación se detenga.

Parámetros:

`newRecordingSpec` – la nueva especificación de grabación se utilizará para reprogramar la RecordingRequest raíz.

Genera:

`java.lang.IllegalArgumentException` – si la nueva especificación de grabación y la especificación de grabación actual para la petición de grabación son diferentes subclases de RecordingSpec.

`AccessDeniedException` – si no se permite que la aplicación llamante realice esta operación mediante atributos de seguridad específicos de RecordingRequest.

`java.lang.SecurityException` – si la aplicación llamante no tiene RecordingPermission("modify",..) or RecordingPermission("*",..)*

---

getId

```
public int getId()
```

Devuelve un identificador para esta petición de grabación. El identificador permitirá identificar de forma inequívoca esta petición de grabación entre todas las que existen en el terminal de grabación GEM. El identificador estará permanentemente asociado con esta petición de grabación en la medida en la misma permanece en el terminal de grabación GEM y, en particular, sobrevivirá a cortes de la alimentación de energía del terminal de grabación GEM. Con ello se consigue que las aplicaciones almacenen dichos ID en sistemas de almacenamiento persistentes para una ulterior recuperación de los mismos por otras aplicaciones u otra instancia de la misma aplicación.

Puesto que las aplicaciones pueden almacenar los identificadores en sistemas persistentes, las implementaciones no deberían reutilizar identificadores de peticiones de grabación que hayan dejado de mantenerse en el terminal de grabación GEM, puesto que ello confundiría a aplicaciones que aún tienen referencias a dichas peticiones de grabación en sus sistemas de almacenamiento persistente.

Devuelve:

Un identificador

Véase también:

`RecordingManager.getRecordingRequest(int)`

D.1.17 Clase RecordingSpec

```
org.ocap.shared.dvr
     Class RecordingSpec

java.lang.Object
  +--org.ocap.shared.dvr.RecordingSpec
```

Direct Known Subclasses:

LocatorRecordingSpec, ServiceContextRecordingSpec, ServiceRecordingSpec

---

clase abstracta pública **RecordingSpec**

amplía java.lang.Object

Clase base para especificar qué grabar y cómo grabarlo.

Resumen del constructor
`RecordingSpec``(RecordingProperties properties)` Constructor.

Resumen del método	
`RecordingProperties`	**`getProperties`**`()` Devuelve la descripción de cómo se hace la grabación.

Métodos heredados de la clase java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Información detallada del constructor

RecordingSpec

`public` **`RecordingSpec`**`(RecordingProperties properties)`

Constructor.

Parámetros:

`properties` – Definición de cómo hacer la descripción.

Información detallada del método

getProperties

```
public RecordingProperties getProperties()
```

Devuelve la descripción de cómo se hace la grabación.

Devuelve:

Las propiedades que han de utilizarse en la grabación.

D.1.18 Clase RecordingTerminatedEvent

```
org.ocap.shared.dvr
     Class RecordingTerminatedEvent

java.lang.Object
  +--java.util.EventObject
       +--javax.tv.service.selection.ServiceContextEvent
            +--org.ocap.shared.dvr.RecordingTerminatedEvent
```

Todas las interfaces implementadas:

java.io.Serializable

---

clase pública **RecordingTerminatedEvent**

amplía javax.tv.service.selection.ServiceContextEvent

Es un evento que notifica que la grabación ha terminado para el `ServiceContext`. Es un evento generado por un `ServiceContext` que está presentando un servicio con desplazamiento temporal o un servicio que está siendo grabado. La presentación no se termina aún ya que el punto de reproducción está retardado en el tiempo. Este evento sólo se genera cuando el punto de reproducción no es el mismo que el punto en vivo. Se generará un `PresentationTerminatedEvent` cuando el punto de reproducción alcance al punto de terminación de la grabación.

Véase también:

Serialized Form

Resumen del campo	
static int	**ACCESS_WITHDRAWN** Código de motivo: el sistema retira el acceso al servicio o algún componente del mismo.
static int	**RESOURCES_REMOVED** Código de motivo: se han eliminado recursos necesarios para grabar el servicio.
static int	**SCHEDULED_STOP** Código de motivo: la grabación terminó normalmente tal como estaba programado.
static int	**SERVICE_VANISHED** Código de motivo: el servicio desapareció de la red.
static int	**USER_STOP** Código de motivo: el usuario solicitó que se detuviera la grabación.

Campos heredados de la clase java.util.EventObject
Source

Resumen del constructor
RecordingTerminatedEvent(javax.tv.service.selection.ServiceContext source, int reason) Construye el evento.

Resumen del método	
int	**getReason**() Devuelve el motivo de terminación de la grabación.

Métodos heredados de la clase javax.tv.service.selection.ServiceContextEvent
getServiceContext

Métodos heredados de la clase java.util.EventObject
getSource, toString

Métodos heredados de la clase java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Información detallada del campo

SERVICE_VANISHED

public static final int **SERVICE_VANISHED**

Código de motivo: el servicio desapareció de la red.

Véase también:

Constant Field Values

RESOURCES_REMOVED

public static final int **RESOURCES_REMOVED**

Código de motivo: se han eliminado recursos necesarios para grabar el servicio. Se generará si se hace una llamada al método de detención ServiceContext.

Véase también:

Constant Field Values

ACCESS_WITHDRAWN

```
public static final int ACCESS_WITHDRAWN
```

Código de motivo: el sistema retira el acceso al servicio o algún componente del mismo. Un ejemplo de ello es el final de un periodo de previsionado gratuito de contenido IPPV.

Véase también:

Constant Field Values

SCHEDULED_STOP

```
public static final int SCHEDULED_STOP
```

Código de motivo: la grabación terminó normalmente tal como estaba programado.

Véase también:

Constant Field Values

USER_STOP

```
public static final int USER_STOP
```

Código de motivo: el usuario solicitó que se detuviera la grabación. También si se hace una llamada al método de detención `RecordingRequest`.

Véase también:

Constant Field Values

Información detallada del constructor

RecordingTerminatedEvent

```
public
RecordingTerminatedEvent(javax.tv.service.selection.ServiceContext source,
                         int reason)
```

Construye el evento.

Parámetros:

`source` – el `ServiceContext` que generó el evento.

Información detallada del método

getReason

```
public int getReason()
```

Devuelve el motivo de terminación de la grabación.

Devuelve:

Motivo de terminación; véanse las constantes de esta clase.

D.1.19 Clase ServiceContextRecordingSpec

```
org.ocap.shared.dvr
     Class ServiceContextRecordingSpec

java.lang.Object
  +--org.ocap.shared.dvr.RecordingSpec
       +--org.ocap.shared.dvr.ServiceContextRecordingSpec
```

---

clase pública **ServiceContextRecordingSpec**

amplía RecordingSpec

Especifica una petición de grabación en los términos en que se está presentando en un contexto de servicio (ServiceContext). Se graban los flujos que están siendo presentados en el parámetro ServiceContext indicado. Si el servicio del que se graba se desintoniza, la grabación SE TERMINARÁ. Si startTime ya ha pasado y la fuente javax.tv.service.selection.ServiceContext está asociada con una memoria intermedia de desplazamiento temporal, el contenido de la memoria intermedia de desplazamiento temporal puede ser inmediatamente almacenado en el destino comenzando, si ello es posible, en el startTime, y llegando hasta el punto de difusión en vivo. Si la memoria intermedia de desplazamiento temporal no contiene la fuente del startTime, podrá grabarse de la fuente tanto como sea posible. Si el startTime ya ha pasado pero no puede asociarse a la grabación una memoria intermedia de desplazamiento temporal, la grabación comienza desde el punto de difusión en vivo. A partir de ese punto, los contenidos de la difusión en vivo se graban durante el tiempo que resta.

Cuando se pasan instancias de este tipo a RecordingManager.record(..), serán de aplicación los modos de fallo adicionales siguientes;

– IllegalArgumentException SERÁ generada si el ServiceContext no está presentando un servicio de difusión o si el startTime es futuro.

– SecurityException SERÁ generada si la aplicación no tiene permiso para acceder al contexto de servicio.

Cuando se pasa como parámetro una instancia de esta especificación de grabación al método RecordingRequest.reschedule(..), se genera una IllegalArgumentException si el parámetro del contexto de servicio es diferente del contexto de servicio especificado en la especificación de grabación actual para la petición de grabación.

Resumen del constructor
ServiceContextRecordingSpec(javax.tv.service.selection.ServiceContext serviceContext, java.util.Date startTime, long duration, RecordingProperties properties) Constructor

Resumen del método	
long	**getDuration**() Devuelve la duración pasada como argumento al constructor.
javax.tv.service.selection.ServiceContext	**getServiceContext**() Devuelve el contexto de servicio (ServiceContext) del que se graba.
java.util.Date	**getStartTime**() Devuelve el tiempo de inicio pasado como argumento al constructor.

Métodos heredados de la clase org.ocap.shared.dvr.RecordingSpec
getProperties

Métodos heredados de la clase java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Información detallada del constructor

ServiceContextRecordingSpec

```
public
ServiceContextRecordingSpec(javax.tv.service.selection.ServiceContext serviceCon
text,
                                    java.util.Date startTime,
                                    long duration,
                                    RecordingProperties properties)
                             throws java.lang.IllegalArgumentException
```

Constructor

Parámetros:

`serviceContext` – ServiceContext del que se graba.

`startTime` – Tiempo de inicio de la grabación. Si el momento de inicio es aún futuro cuando se llama al método RecordingManager.record(..) con este ServiceContextRecordingSpec como argumento, el método de grabación generará una IllegalArgumentException.

`duration` – duración de la grabación en milisegundos.

`properties` – definición de cómo debe hacerse la grabación

Genera:

`java.lang.IllegalArgumentException` – si la duración es negativa

Información detallada del método

getServiceContext

```
public javax.tv.service.selection.ServiceContext getServiceContext()
```

Devuelve el ServiceContext del que se graba

Devuelve:

La instancia de ServiceContext pasada al constructor.

---

getStartTime

```
public java.util.Date getStartTime()
```

Devuelve el tiempo de inicio pasado como un argumento al constructor.

Devuelve:

El tiempo de inicio pasado al constructor.

---

getDuration

```
public long getDuration()
```

Devuelve la duración pasada como argumento al constructor.

Devuelve:

La duración pasada al constructor.

D.2 Paquete de navegación de registro vídeo numérico compartido

```
org.ocap.shared.dvr.navigation
     Package
```


D.2.1 Clase RecordingStateFilter

```
org.ocap.shared.dvr.navigation
     Class RecordingStateFilter

java.lang.Object
  +--org.ocap.shared.dvr.navigation.RecordingListFilter
       +--org.ocap.shared.dvr.navigation.RecordingStateFilter
```

---

clase pública **RecordingStateFilter**

amplía RecordingListFilter

Filtro para filtrado basado en valores devueltos por el método getState en `RecordingRequest`.

Resumen del constructor
`RecordingStateFilter``(int state)` Construye el filtro basado en un tipo de estado particular (PENDING, FAILED, etc.).

Resumen del método	
`boolean`	**`accept`**`(RecordingRequest entry)` Verifica si el `RecordingRequest` dado pasa el filtro.
`int`	**`getFilterValue`**`()` Informa del valor del estado utilizado para crear este filtro.

Métodos heredados de la clase org.ocap.shared.dvr.navigation.RecordingListFilter
`setCascadingFilter`

Métodos heredados de la clase java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Información detallada del constructor

RecordingStateFilter

`public` **`RecordingStateFilter`**`(int state)`

Construye el filtro basado en un tipo de estado particular (PENDING, FAILED, etc.).

Parámetros:

`state` – Valor que corresponde con el estado de una instancia de `RecordingRequest`.

Información detallada del método

getFilterValue

`public int` **`getFilterValue`**`()`

Informa del valor del estado utilizado para crear este filtro.

Devuelve:

El valor del estado utilizado para crear este filtro..

accept

`public boolean accept(RecordingRequest entry)`

Verifica si el `RecordingRequest` dado pasa el filtro.

Especificado por:

`accept` en la clase `RecordingListFilter`

Parámetros:

`entry` – una RecordingRequest individual que ha de ser evaluada según el algoritmo de filtrado.

Devuelve:

`true` si el `RecordingRequest` incluido en el parámetro RecordingRequest se encuentra en el estado indicado por el valor del filtro; en cualquier otro caso, `false`.

D.2.2 Clase AppIDFilter

```
org.ocap.shared.dvr.navigation
     Class AppIDFilter

java.lang.Object
  +--org.ocap.shared.dvr.navigation.RecordingListFilter
       +--org.ocap.shared.dvr.navigation.AppIDFilter
```

---

clase pública **AppIDFilter**

amplía RecordingListFilter

Filtro para filtrado basado en AppID.

Resumen del constructor
AppIDFilter(org.dvb.application.AppID appID) Construye el filtro basado en un `AppID` específico.

Resumen del método	
boolean	**accept**(RecordingRequest entry) Verifica si el `RecordingRequest` dado pasa el filtro.
org.dvb.application.AppID	**getFilterValue**() Informa del valor de `AppID` utilizado para crear este filtro.

Métodos heredados de la clase org.ocap.shared.dvr.navigation.RecordingListFilter
setCascadingFilter

Métodos heredados de la clase java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Información detallada del constructor

AppIDFilter

`public` **`AppIDFilter`**`(org.dvb.application.AppID appID)`

Construye el filtro basado en un `AppID` específico.

Parámetros:

`appID` – valor de AppID que corresponde con las `RecordingRequest`s.

Información detallada del método

getFilterValue

`public org.dvb.application.AppID` **`getFilterValue`**`()`

Informa del valor de `AppID` utilizado para crear este filtro.

Devuelve:

El valor de `AppID` utilizado para crear este filtro.

---

accept

`public boolean` **`accept`**`(RecordingRequest entry)`

Verifica si el `RecordingRequest` dado pasa el filtro.

Especificado por:

`accept` en la clase `RecordingListFilter`

Parámetros:

`entry` – Una `RecordingRequest` individual que ha de ser evaluada según el algoritmo de filtrado.

Devuelve:

`true` si `RecordingRequest` es del tipo indicado por el valor del filtro; en cualquier otro caso, `false`.

D.2.3 Clase OrgIDFilter

```
org.ocap.shared.dvr.navigation
     Class OrgIDFilter

java.lang.Object
  +--org.ocap.shared.dvr.navigation.RecordingListFilter
       +--org.ocap.shared.dvr.navigation.OrgIDFilter
```

---

clase pública **OrgIDFilter**

amplía RecordingListFilter

Filtro para filtrado basado en OrgID.

Resumen del constructor
`OrgIDFilter``(int orgID)` Construye el filtro basado en un ID de organización específico.

Resumen del método	
`boolean`	**`accept`**`(RecordingRequest entry)` Verifica si el `RecordingRequest` dado pasa el filtro.
`int`	**`getFilterValue`**`()` Informa del valor del ID de organización utilizado para crear este filtro.

Métodos heredados de la clase org.ocap.shared.dvr.navigation.RecordingListFilter
`setCascadingFilter`

Métodos heredados de la clase java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Información detallada del constructor

OrgIDFilter

`public` **`OrgIDFilter`**`(int orgID)`

Construye el filtro basado en un ID de organización específico.

Parámetros:

`orgID` – valor del ID de organización que corresponde con las instancias de `RecordingRequest`.

Información detallada del método

getFilterValue

`public int` **`getFilterValue`**`()`

Informa del valor del ID de organización utilizado para crear este filtro.

Devuelve:

El ID de la organización utilizado para el filtro.

accept

`public boolean` **`accept`**`(RecordingRequest entry)`

Verifica si el `RecordingRequest` dado pasa el filtro.

Especificado por:

`accept` en la clase `RecordingListFilter`

Parámetros:

`entry` – Una `RecordingRequest` individual que ha de ser evaluada según el algoritmo de filtrado.

Devuelve:

`true` si el `RecordingRequest` es del tipo indicado por el valor del filtro; en cualquier otro caso, `false`.

D.2.4 Interfaz RecordingList

```
org.ocap.shared.dvr.navigation
     Interface RecordingList
```

---

interfaz pública **RecordingList**

RecordingList representa una lista de grabaciones.

Resumen del método	
`boolean`	**`contains`**`(RecordingRequest entry)` Verifica si el objeto `RecordingRequest` indicado está incluido en la lista.
`RecordingListIterator`	**`createRecordingListIterator`**`()` Genera un iterador de los elementos `RecordingRequest` de la lista.
`RecordingList`	**`filterRecordingList`**`(RecordingListFilter filter)` Crea un nuevo objeto `RecordingList` que es un subconjunto de esta lista, basado en las condiciones especificadas por un objeto `RecordingListFilter`.
`RecordingRequest`	**`getRecordingRequest`**`(int index)` Informa del `RecordingRequest` en la posición índice especificada.
`int`	**`indexOf`**`(RecordingRequest entry)` Informa de la posición de la primera ocurrencia en la lista del objeto `RecordingRequest` indicado.
`int`	**`size`**`()` Informa del número de objetos `RecordingRequest` en la lista.
`RecordingList`	**`sortRecordingList`**`(RecordingListComparator sortCriteria)` Crea un nuevo objeto `RecordingList` que contiene todos los elementos de la lista ordenados de acuerdo con los criterios especificados por un `RecordingListComparator`.

Información detallada del método

filterRecordingList

```
public RecordingList filterRecordingList(RecordingListFilter filter)
```

Crea un nuevo objeto `RecordingList` que es un subconjunto de esta lista, basado en las condiciones especificadas por un objeto `RecordingListFilter`. Este método puede utilizarse para generar listas cada vez más especializadas de objetos `RecordingRequest` en base a múltiples criterios de filtrado. Si el filtro es `null`, el `RecordingList` resultante será un duplicado de la lista.

Obsérvese que el método `accept` del `RecordingListFilter` dado, será invocado para cada `RecordingRequest` que deba ser filtrada utilizando la misma secuencia de aplicación que invoca este método.

Parámetros:

`filter` – Un filtro que limita la lista de grabaciones solicitadas, o `null`.

Devuelve:

Un objeto `RecordingList` creado en base a reglas de filtrado especificadas.

---

createRecordingListIterator

```
public RecordingListIterator createRecordingListIterator()
```

Genera un iterador de los elementos `RecordingRequest` de la lista.

Devuelve:

Un `RecordingListIterator` sobre las `RecordingRequest`s de la lista.

---

contains

```
public boolean contains(RecordingRequest entry)
```

Verifica si el objeto `RecordingRequest` indicado está incluido en la lista.

Parámetros:

`entry` – El objeto `RecordingRequest` buscado.

Devuelve:

`true` si el `RecordingRequest` es miembro de la lista; en cualquier otro caso, `false`.

---

indexOf

```
public int indexOf(RecordingRequest entry)
```

Informa de la posición de la primera ocurrencia en la lista del objeto `RecordingRequest` indicado.

Parámetros:

`entry` – El objeto `RecordingRequest` buscado.

Devuelve:

El índice de la primera ocurrencia de `entry`, ó `-1` si dicha entrada no está incluida en la lista.

---

size

```
public int size()
```

Informa del número de objetos `RecordingRequest` en la lista

Devuelve:

El número de objetos `RecordingRequest` en la lista.

---

getRecordingRequest

```
public RecordingRequest getRecordingRequest(int index)
```

Informa del `RecordingRequest` en la posición índice especificada

Parámetros:

`index` – Una posición en la `RecordingList`.

Devuelve:

El `RecordingRequest` en el índice especificado.

Genera:

`java.lang.IndexOutOfBoundsException` – Si `index` < 0 ó `index` > `size()-1`.

---

sortRecordingList

```
public RecordingList sortRecordingList(RecordingListComparator sortCriteria)
```

Crea un nuevo objeto `RecordingList` que contiene todos los elementos de la lista ordenados de acuerdo con los criterios especificados por un `RecordingListComparator`

Parámetros:

`sortCriteria` – el criterio de ordenación a utilizar para ordenar las entradas de la lista de grabaciones.

Devuelve:

Una copia ordenada de la lista de grabaciones.

D.2.5 Interfaz RecordingListComparator

```
org.ocap.shared.dvr.navigation
     Interface RecordingListComparator
```

---

interfaz pública **RecordingListComparator**

Esta interfaz representa un criterio de ordenación aplicado para ordenar una RecordingList.

Resumen del método	
int	**compare**(RecordingRequest first, RecordingRequest second) Compara dos entradas para verificar si la primera debe situarse antes que la segunda en la lista de iteración.

Información detallada del método

compare

```
public int compare(RecordingRequest first,
                   RecordingRequest second)
```

Compara dos entradas para verificar si la primera debe situarse antes que la segunda en la lista de iteración.

Parámetros:

`first` – primera entrada de la comparación

`second` – segunda entrada de la comparación

Devuelve:

Un entero positivo si el primer argumento debe colocarse antes que el segundo; un entero negativo si el segundo argumento debe colocarse antes que el primero; cero si debe mantenerse el orden actual.

D.2.6 Clase RecordingListFilter

```
org.ocap.shared.dvr.navigation
     Class RecordingListFilter

java.lang.Object
  +--org.ocap.shared.dvr.navigation.RecordingListFilter
```

Direct Known Subclasses:

AppIDFilter, OrgIDFilter, RecordingStateFilter

---

clase abstracta pública **RecordingListFilter**

amplía java.lang.Object

Clase base para todos los RecordingListFilters. Pueden utilizarse subclases de RecordingListFilter para crear filtros que especifiquen restricciones.

Resumen del constructor	
protected	**RecordingListFilter**() Construye el filtro.

Resumen del método	
abstract boolean	**accept**(RecordingRequest entry) Verifica si una entrada en particular pasa este filtro.
void	**setCascadingFilter**(RecordingListFilter filter) Permite aplicar los filtros en tándem.

Métodos heredados de la clase java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Información detallada del constructor

RecordingListFilter

protected **RecordingListFilter**()

Construye el filtro.

Información detallada del método

accept

public abstract boolean **accept**(RecordingRequest entry)

Verifica si una entrada en particular pasa este filtro. Los subtipos de `RecordingListFilter` no cumplen este método para proporcionar la lógica de una operación de filtrado sobre objetos `RecordingRequest` individuales.

Parámetros:

`entry` – Un `RecordingRequest` que ha de ser evaluado según el algoritmo de filtrado.

Devuelve:

`true` si `entry` cumple el algoritmo de filtrado; en cualquier otro caso, `false`.

---

setCascadingFilter

public void **setCascadingFilter**(RecordingListFilter filter)

Permite aplicar los filtros en tándem. El método accept de este filtro sólo se llama en el caso de entradas que concuerden con el filtro especificado. Varias llamadas a este método sustituyen al filtro anteriormente fijado.

Parámetros:

`filter` – filtro que se está aplicando antes de seleccionar las entradas para las que se llama al método accept(). Si el filtro actual pertenece a la cadena en tándem del filtro pasado como argumento, este método no hace nada.

D.2.7 Interfaz RecordingListIterator

```
org.ocap.shared.dvr.navigation
     Interface RecordingListIterator
```

---

interfaz pública **RecordingListIterator**

Este iterador puede utilizarse para examinar y desplazarse a lo largo de las anotaciones de una RecordingList.

Resumen del método	
`RecordingRequest`	**`getEntry`**`(int index)` Obtiene el objeto `RecordingRequest` que ocupa la posición especificada.
`int`	**`getPosition`**`()` Obtiene la posición actual del RecordingListIterator.
`int`	**`getPosition`**`(RecordingRequest entry)` Obtiene la posición en la lista de la petición de grabación especificada.
`RecordingList`	**`getRecordingList`**`()` Obtiene la lista de grabación correspondiente a este RecordingListIterator.
`boolean`	**`hasNext`**`()` Verifica si hay un `RecordingRequest` en la posición siguiente de la lista.
`boolean`	**`hasPrevious`**`()` Verifica si hay un `RecordingRequest` en la posición anterior de la lista.
`RecordingRequest[]`	**`nextEntries`**`(int n)` Obtiene los siguientes 'n' objetos `RecordingRequest` de la lista.
`RecordingRequest`	**`nextEntry`**`()` Obtiene el siguiente objeto `RecordingRequest` de la lista.
`RecordingRequest[]`	**`previousEntries`**`(int n)` Obtiene los 'n' objetos `RecordingRequest` anteriores de la lista.
`RecordingRequest`	**`previousEntry`**`()` Obtiene el objeto `RecordingRequest` anterior de la lista.
`void`	**`setPosition`**`(int index)` Establece la posición actual del RecordingListIterator.
`void`	**`toBeginning`**`()` Reinicia el iterador al comienzo de la lista, de forma que `hasPrevious()` devuelve `false` y `nextEntry()` devuelve el primer `RecordingRequest` de la lista (si no está vacía).
`void`	**`toEnd`**`()` Fija el iterador al final de la lista, de forma que `hasNext()` devuelve `false` y `previousEntry()` devuelve el último `RecordingRequest` de la lista (si no está vacía).

Información detallada del método

toBeginning

`public void **toBeginning**()`

Reinicia el iterador al comienzo de la lista, de forma que `hasPrevious()` devuelve `false` y `nextEntry()` devuelve el primer `RecordingRequest` de la lista (si no está vacía).

---

toEnd

`public void **toEnd**()`

Fija el iterador al final de la lista, de forma que `hasNext()` devuelve `false` y `previousEntry()` devuelve el último `RecordingRequest` de la lista (si no está vacía).

---

nextEntry

`public RecordingRequest **nextEntry**()`

Obtiene el siguiente objeto `RecordingRequest` de la lista. Se puede llamar repetidamente a este método para realizar iteraciones en la lista.

Devuelve:

El objeto `RecordingRequest` de la siguiente posición de la lista.

Genera:

`java.util.NoSuchElementException` – Si la iteración no presenta a continuación un `RecordingRequest`.

---

previousEntry

`public RecordingRequest **previousEntry**()`

Obtiene el objeto `RecordingRequest` anterior de la lista. Se puede llamar repetidamente a este método para realizar iteraciones en orden inverso en la lista.

Devuelve:

El objeto `RecordingRequest` de la posición anterior de la lista.

Genera:

`java.util.NoSuchElementException` – Si la iteración no presenta un `RecordingRequest` anterior.

---

hasNext

`public boolean **hasNext**()`

Verifica si hay un `RecordingRequest` en la posición siguiente de la lista.

Devuelve:

`true` si hay un `RecordingRequest` en la posición siguiente de la lista; en cualquier otro caso, `false`.

---

hasPrevious

`public boolean **hasPrevious**()`

Verifica si hay un `RecordingRequest` en la posición anterior de la lista.

Devuelve:

`true` si hay un `RecordingRequest` en la posición anterior de la lista; en cualquier otro caso, `false`.

---

nextEntries

`public RecordingRequest[] **nextEntries**(int n)`

Obtiene los siguientes 'n' objetos `RecordingRequest` de la lista. Este método también permite conocer la posición actual en la lista. Si no está disponible el número solicitado de entradas, se devuelven los restantes elementos. Si la posición actual está al final del iterador, este método devuelve una matriz de longitud cero.

Parámetros:

`n` – el número de las siguientes entradas solicitadas.

Devuelve:

Una matriz que contiene los siguientes 'n' objetos `RecordingRequest` a partir de la posición actual en la lista.

---

previousEntries

`public RecordingRequest[] **previousEntries**(int n)`

Obtiene los 'n' objetos `RecordingRequest` anteriores de la lista. Este método también modifica la posición actual en la lista. Si no están disponibles el número solicitado de entradas, se devuelven los restantes elementos. Si la posición actual está al comienzo del iterador, este método devuelve una matriz de longitud cero.

Parámetros:

`n` – el número de las entradas anteriores solicitadas.

Devuelve:

Una matriz que contiene los 'n' objetos `RecordingRequest` anteriores a partir de la posición actual en la lista.

---

getEntry

`public RecordingRequest **getEntry**(int index)`

Obtiene el objeto `RecordingRequest` que ocupa la posición especificada. Este método no permite conocer cuál es la posición actual en la lista.

Parámetros:

`index` – posición del `RecordingRequest` que debe recuperarse.

Devuelve:

El `RecordingRequest` en la posición especificada.

Genera:

`java.lang.IndexOutOfBoundsException` – si el índice es superior al tamaño de la lista.

---

getPosition

```
public int getPosition(RecordingRequest entry)
```

Obtiene la posición en la lista de la petición de grabación especificada.

Parámetros:

`entry` – petición de grabación cuya posición desea conocerse.

Devuelve:

la posición de la grabación especificada; –1 si no se encuentra la entrada.

---

getPosition

```
public int getPosition()
```

Obtiene la posición actual del RecordingListIterator. Sería la posición a partir de la cual se recupera el siguiente RecordingRequest cuando una aplicación llama a nextEntry.

Devuelve:

La posición actual del RecordingListIterator.

---

setPosition

```
public void setPosition(int index)
                 throws java.lang.IndexOutOfBoundsException
```

Establece la posición actual del RecordingListIterator. Sería la posición a partir de la cual se recupera el siguiente RecordingRequest cuando una aplicación llama a nextEntry.

Parámetros:

`index` – valor que tomaría la posición actual del RecordingListIterator.

Genera:

`java.lang.IndexOutOfBoundsException` – si el índice es superior al tamaño de la lista.

---

getRecordingList

```
public RecordingList getRecordingList()
```

Obtiene la lista de grabación correspondiente a este RecordingListIterator.

Devuelve:

El RecordingList correspondiente a este iterador.

D.3 Paquete de medios compartido

```
org.ocap.shared.media
     Package
```


D.3.1 Interfaz TimeShiftControl

```
org.ocap.shared.media
     Interface TimeShiftControl
```

Todas las superinterfaces:

javax.media.Control

---

interfaz pública **TimeShiftControl**

amplía javax.media.Control

Esta interfaz representa un control del modo truco que puede utilizarse para recuperar más información sobre la reproducción de la memoria intermedia de desplazamiento temporal. Este control sólo estará disponible si el servicio presentado en el contexto del servicio es un servicio de difusión y si existe una memoria intermedia de desplazamiento temporal asociada a dicho contexto de servicio.

Resumen del método	
`javax.media.Time`	**`getBeginningOfBuffer`**`()` Obtiene el tiempo de medios correspondiente al comienzo actual de la memoria intermedia de desplazamiento temporal.
`javax.media.Time`	**`getDuration`**`()` Obtiene la duración del contenido que se encuentra actualmente en la memoria intermedia de desplazamiento temporal.
`javax.media.Time`	**`getEndOfBuffer`**`()` Obtiene el tiempo de medios correspondiente al final de la memoria intermedia de desplazamiento temporal.
`javax.media.Time`	**`getMaxDuration`**`()` Obtiene el valor estimado de la duración máxima del contenido que podría almacenarse utilizando dicha memoria intermedia de desplazamiento temporal.

Métodos heredados de la interfaz javax.media.Control
`getControlComponent`

Información detallada del método

getBeginningOfBuffer

```
public javax.media.Time getBeginningOfBuffer()
```

Obtiene el tiempo de medios correspondiente al comienzo actual de la memoria intermedia de desplazamiento temporal. Podría ser el tiempo de medios correspondiente al comienzo de la memoria intermedia, antes de que ésta comience a sobreescribirse de nuevo, o el tiempo de medios correspondiente al comienzo del área de memoria intermedia válida a continuación de la parte sobreescrita.

Devuelve:

El tiempo de medios correspondiente al comienzo de la memoria intermedia de desplazamiento temporal.

---

getEndOfBuffer

```
public javax.media.Time getEndOfBuffer()
```

Obtiene el tiempo de medios correspondiente al final de la memoria intermedia de desplazamiento temporal. Podría ser el tiempo actual del sistema si la grabación con desplazamiento temporal está aún en curso o el tiempo de medios correspondiente al último punto de la zona válida de la memoria intermedia de desplazamiento temporal.

Devuelve:

Tiempo de medios correspondiente al final de la memoria intermedia de desplazamiento temporal.

---

getDuration

```
public javax.media.Time getDuration()
```

Obtiene la duración del contenido que se encuentra actualmente en la memoria intermedia de desplazamiento temporal. El valor devuelto es la duración del contenido cuando se reproduce a la velocidad de 1,0.

Devuelve:

Un objeto Time que representa la duración.

---

getMaxDuration

```
public javax.media.Time getMaxDuration()
```

Obtiene el valor estimado de la duración máxima del contenido que podría almacenarse utilizando dicha memoria intermedia de desplazamiento temporal. El valor devuelto es la duración del contenido cuando se reproduce a la velocidad de 1,0.

Devuelve:

Un objeto Time que representa la duración máxima.

D.3.2 Clase BeginningOfContentEvent

```
org.ocap.shared.media
     Class BeginningOfContentEvent

java.lang.Object
  +--java.util.EventObject
       +--javax.media.ControllerEvent
            +--javax.media.RateChangeEvent
                 +--org.ocap.shared.media.BeginningOfContentEvent
```

Todas las interfaces implementadas:

javax.media.MediaEvent, java.io.Serializable

---

clase pública **BeginningOfContentEvent**

amplía javax.media.RateChangeEvent

BeginningOfContentEvent (evento comienzo de contenido) es un RateChangeEvent (evento cambio de velocidad) que se inserta cuando el cambio de velocidad se debe a que en un retroceso rápido se alcanza el comienzo de medios o porque la ocupación de la memoria intermedia de desplazamiento temporal alcanza su capacidad máxima.

Véase también:

Serialized Form

Resumen del campo

Campos heredados de la clase java.util.EventObject
`Source`

Resumen del constructor
`BeginningOfContentEvent``(javax.media.Controller from, float newRate)` Crea un BeginningOfContentEvent.

Métodos heredados de la clase javax.media.RateChangeEvent
`getRate, toString`

Métodos heredados de la clase javax.media.ControllerEvent
`getSource, getSourceController`

Métodos heredados de la clase java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Información detallada del constructor

BeginningOfContentEvent

```
public BeginningOfContentEvent(javax.media.Controller from,
                               float newRate)
```

Crea un BeginningOfContentEvent.

Parámetros:

`from` – controlador que genera el evento.

D.3.3 Clase EndOfContentEvent

```
org.ocap.shared.media
     Class EndOfContentEvent

java.lang.Object
  +--java.util.EventObject
      +--javax.media.ControllerEvent
          +--javax.media.RateChangeEvent
              +--org.ocap.shared.media.EndOfContentEvent
```

Todas las interfaces implementadas:

javax.media.MediaEvent, java.io.Serializable

---

clase pública **EndOfContentEvent**

amplía javax.media.RateChangeEvent

EndOfContentEvent (evento final de contenido) es un RateChangeEvent (evento cambio de velocidad) que se inserta cuando el cambio de velocidad se debe que en un avance rápido se alcanza el punto de final del contexto almacenado o que el avance rápido alcanza el punto en el punto de la grabación en directo.

Véase también:

Serialized Form

Resumen del campo

Campos heredados de la clase java.util.EventObject

source

Resumen del constructor

EndOfContentEvent(javax.media.Controller from, float newRate) Crea un EndOfContentEvent.

Métodos heredados de la clase javax.media.RateChangeEvent
getRate, toString

Métodos heredados de la clase javax.media.ControllerEvent
getSource, getSourceController

Métodos heredados de la clase java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Información detallada del constructor

EndOfContentEvent

```
public EndOfContentEvent(javax.media.Controller from,
                         float newRate)
```

Crea un EndOfContentEvent.

Parámetros:

`from` – controlador que genera el evento.

D.3.4 Clase EnteringLiveModeEvent

```
org.ocap.shared.media
     Class EnteringLiveModeEvent

java.lang.Object
  +--java.util.EventObject
       +--javax.media.ControllerEvent
            +--org.ocap.shared.media.EnteringLiveModeEvent
```

Todas las interfaces implementadas:

javax.media.MediaEvent, java.io.Serializable

---

clase pública **EnteringLiveModeEvent**

amplía javax.media.ControllerEvent

EnteringLiveModeEvent (evento de paso a modo en directo) es un ControllerEvent (evento controlador) que se inserta cuando el controlador ha comenzado la reproducción de un flujo de difusión en directo. Este evento se envía a un ControllerListener registrado junto a los RateChangeEvent o MediaTimeSetEvent.

Véase también:

Serialized Form

Resumen del campo

Campos heredados de la clase java.util.EventObject

`source`

Resumen del constructor

`EnteringLiveModeEvent``(javax.media.Controller from)`
Crea un EnteringLiveModeEvent.

Métodos heredados de la clase javax.media.ControllerEvent

`getSource, getSourceController, toString`

Métodos heredados de la clase java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Información detallada del constructor

EnteringLiveModeEvent

`public` **`EnteringLiveModeEvent`**`(javax.media.Controller from)`

Crea un EnteringLiveModeEvent.

Parámetros:

`from` – controlador que genera el evento.

D.3.5 Clase LeavingLiveModeEvent

```
org.ocap.shared.media
     Class LeavingLiveModeEvent

java.lang.Object
  +--java.util.EventObject
       +--javax.media.ControllerEvent
            +--org.ocap.shared.media.LeavingLiveModeEvent
```

Todas las interfaces implementadas:

javax.media.MediaEvent, java.io.Serializable

---

clase pública **LeavingLiveModeEvent**

amplía javax.media.ControllerEvent

LeavingLiveModeEvent (evento abandono de modo en directo) es un ControllerEvent (evento controlador) que se inserta cuando el controlador deja de reproducir definitivamente un flujo de difusión en directo. Este evento se envía a un ControllerListener registrado junto a los RateChangeEvent o MediaTimeSetEvent.

Véase también:

Serialized Form

Resumen del campo

Campos heredados de la clase java.util.EventObject
`source`

Resumen del constructor
`LeavingLiveModeEvent``(javax.media.Controller from)` Crea un LeavingLiveModeEvent.

Métodos heredados de la clase javax.media.ControllerEvent
`getSource, getSourceController, toString`

Métodos heredados de la clase java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Información detallada del constructor

LeavingLiveModeEvent

`public` **`LeavingLiveModeEvent`**`(javax.media.Controller from)`

Crea un LeavingLiveModeEvent.

Parámetros:

`from` – controlador que genera el evento.

D.3.6 Interfaz MediaTimeFactoryControl

```
org.ocap.shared.media
     Interface MediaTimeFactoryControl
```

Todas las superinterfaces:

javax.media.Control

---

interfaz pública **MediaTimeFactoryControl**

amplía javax.media.Control

Permite obtener tiempos de medios con diversas características especiales cuando se aplica a contenidos que están siendo reproducidos por el reproductor JMF. El comportamiento de dichos tiempos de medios es función de la implementación si se utilizan con cualquier otro reproductor JMF.

Resumen del método	
javax.media.Time	**getRelativeTime**(long offset) Obtiene un tiempo de medios relativo a la ubicación actual.
javax.media.Time	**setTimeApproximations**(javax.media.Time original, boolean beforeOrAfter) Permite que las aplicaciones controlen la posición en la que comienza la reproducción después de haberse hecho una llamada a Player.setMediaTime.

Métodos heredados de la interfaz javax.media.Control
getControlComponent

Información detallada del método

getRelativeTime

```
public javax.media.Time getRelativeTime(long offset)
```

Obtiene un tiempo de medios relativo a la ubicación actual.

Parámetros:

`offset` – desplazamiento relativo a la posición actual medido en nanosegundos.

Devuelve:

Un tiempo de medios.

---

setTimeApproximations

```
public javax.media.Time setTimeApproximations(javax.media.Time original,
                                              boolean beforeOrAfter)
```

Permite que las aplicaciones controlen con exactitud la posición en la que comienza la reproducción después de una llamada a Player.setMediaTime. Este método toma como entrada un tiempo de medios original y devuelve un nuevo tiempo de medios que encapsula el tiempo de medios original y una indicación de cómo debe interpretarse dicho tiempo de medios original cuando se utiliza en una llamada a Player.setMediaTime.

Parámetros:

`original` – tiempo de medios original

`beforeOrAfter` – si es verdadero, el tiempo de medios en el que comienza la reproducción será anterior o coincidirá con el tiempo de medios original (es decir, se garantiza que se incluya en la reproducción el contenido existente en el tiempo de medios original). Si es falso, el tiempo de medios en el que se inicia la reproducción será posterior al original (es decir, la reproducción no incluirá ni el contenido existente en el tiempo de medios original ni ningún contenido anterior al mismo).

Devuelve:

Un nuevo tiempo de medios.

D.3.7 Interfaz TimeLine

```
org.ocap.shared.media
     Interface TimeLine
```

---

interfaz pública **TimeLine**

Representa una secuencia temporal transmitida. Las secuencias temporales transmitidas comienzan en un tiempo de medios dentro de un contenido y finalizan en un tiempo de medios posterior de dicho contenido. Las secuencias temporales transmitidas son válidas en todos los tiempos de medios que existan entre dichos dos puntos. Pueden tener un crecimiento lineal o sufrir pausas. El valor de una secuencia temporal transmitida no presenta discontinuidades.

Resumen del método	
`javax.media.Time`	**`getFirstMediaTime`**`()` Devuelve el primer tiempo de medios en el que la secuencia temporal es válida.
`long`	**`getFirstTime`**`()` Devuelve el primer tiempo válido de la secuencia temporal.
`javax.media.Time`	**`getLastMediaTime`**`()` Devuelve el último tiempo en que la secuencia temporal es válida.
`long`	**`getLastTime`**`()` Devuelve el último tiempo válido de la secuencia temporal.
`javax.media.Time`	**`getMediaTime`**`(long time)` Traduce un tiempo de la secuencia temporal en el correspondiente tiempo de medios.
`long`	**`getTime`**`(javax.media.Time mediatime)` Traduce un tiempo de medios en el correspondiente tiempo en la secuencia temporal.

Información detallada del método

getFirstMediaTime

```
public javax.media.Time getFirstMediaTime()
                                  throws TimeLineInvalidException
```

Devuelve el primer tiempo de medios en el que la secuencia temporal es válida. Para una grabación programada, se trata del primer punto dentro del contenido en que la secuencia temporal es válida. En el caso de una grabación con desplazamiento temporal, si la secuencia temporal comienza dentro de la memoria intermedia de desplazamiento temporal, se devuelve el tiempo de medios en que aquélla comienza. Si la secuencia temporal comienza antes del inicio de la memoria intermedia de desplazamiento temporal, se devuelve el tiempo de medios en el que comienza la memoria intermedia de desplazamiento temporal. Obsérvese que si ésta está llena y hay en curso una grabación con desplazamiento temporal, el comienzo de la memoria intermedia se desplazará conforme se sobreescriban nuevos datos sobre el anterior comienzo de la memoria intermedia.

Devuelve:

Un tiempo de medios.

Genera:

`TimeLineInvalidException` – si la secuencia temporal deja de ser válida en el contenido en cuestión, por ejemplo, porque el contenido sea una grabación con desplazamiento temporal y el final de la secuencia temporal ya no esté dentro de la memoria intermedia.

---

getLastMediaTime

```
public javax.media.Time getLastMediaTime()
                                  throws TimeLineInvalidException
```

Devuelve el último tiempo en que la secuencia temporal es válida. Para una grabación programada, se trata del último punto dentro del contenido en que la secuencia temporal es válida. En el caso de una grabación con desplazamiento temporal, si la secuencia temporal termina dentro de la memoria intermedia de desplazamiento temporal, se devuelve el tiempo de medios en que aquélla comienza. Si la secuencia temporal termina después del final de la memoria intermedia de desplazamiento temporal, se devuelve el tiempo de medios del final de la memoria intermedia de desplazamiento temporal. Obsérvese que si ésta está llena y hay en curso una grabación con desplazamiento temporal, el final de la memoria intermedia se desplazará conforme se sobreescriban nuevos datos sobre el anterior comienzo de la memoria intermedia.

Devuelve:

Un tiempo de medios.

Genera:

`TimeLineInvalidException` – si la secuencia temporal deja de ser válida en el contenido en cuestión, por ejemplo, porque el contenido sea una grabación con desplazamiento temporal y el final de la secuencia temporal ya no esté dentro de la memoria intermedia.

---

getFirstTime

```
public long getFirstTime()
                  throws TimeLineInvalidException
```

Devuelve el primer tiempo válido de la secuencia temporal. Para una grabación programada, se trata del primer punto dentro del contenido en que la secuencia temporal es válida. En el caso de una grabación con desplazamiento temporal, si la secuencia temporal comienza dentro de la memoria intermedia de desplazamiento temporal, se devuelve el tiempo en que aquélla comienza. Si la secuencia temporal comienza antes del inicio de la memoria intermedia de desplazamiento temporal, se devuelve el tiempo de comienzo de la memoria intermedia de desplazamiento temporal. Obsérvese que si ésta está llena y hay en curso una grabación con desplazamiento temporal, el comienzo de la memoria intermedia se desplazará conforme se sobreescriban nuevos datos sobre el anterior comienzo de la memoria intermedia.

Devuelve:

Un tiempo incluido en la secuencia temporal.

Genera:

`TimeLineInvalidException` – si la secuencia temporal deja de ser válida en el contenido en cuestión, por ejemplo, porque el contenido sea una grabación con desplazamiento temporal y el final de la secuencia temporal ya no esté dentro de la memoria intermedia.

---

getLastTime

```
public long getLastTime()
                  throws TimeLineInvalidException
```

Devuelve el último tiempo válido de la secuencia temporal. Para una grabación programada, se trata del último punto dentro del contenido en que la secuencia temporal es válida. En el caso de una grabación con desplazamiento temporal, si la secuencia temporal termina dentro de la memoria intermedia de desplazamiento temporal, se devuelve el tiempo de medios en que aquélla comienza. Si la secuencia temporal termina después del final de la memoria intermedia de desplazamiento temporal, se devuelve el tiempo de medios en que finaliza la memoria intermedia de desplazamiento temporal. Obsérvese que si ésta está llena y hay en curso una grabación con desplazamiento temporal, el final de la memoria intermedia se desplazará conforme se sobreescriban nuevos datos sobre el anterior inicio de la memoria intermedia.

Devuelve:

Un tiempo incluido en la secuencia temporal.

Genera:

`TimeLineInvalidException` – si la secuencia temporal deja de ser válida en el contenido en cuestión, por ejemplo, porque el contenido sea una grabación con desplazamiento temporal y el final de la secuencia temporal ya no está dentro de la memoria intermedia

---

getMediaTime

```
public javax.media.Time getMediaTime(long time)
                                throws TimeLineInvalidException,
                                       TimeOutOfRangeException
```

Traduce un tiempo de la secuencia temporal en el correspondiente tiempo de medios. Si el tiempo coincide con uno en que la secuencia temporal presenta una pausa, el tiempo de medios devuelto será el mayor tiempo de medios del tiempo especificado.

Parámetros:

`time` – un tiempo incluido en la secuencia temporal

Devuelve:

El tiempo de medios correspondiente.

Genera:

`TimeLineInvalidException` – si la secuencia temporal deja de ser válida en el contenido en cuestión, por ejemplo, porque el contenido sea una grabación con desplazamiento temporal y el final de la secuencia temporal ya no está dentro de la memoria intermedia

`TimeOutOfRangeException` – si el tiempo especificado no está dentro de la secuencia temporal.

---

getTime

```
public long getTime(javax.media.Time mediatime)
             throws TimeLineInvalidException,
                    TimeOutOfRangeException
```

Traduce un tiempo de medios en el correspondiente tiempo de la secuencia temporal.

Parámetros:

`mediatime` – un tiempo de medios

Devuelve:

El correspondiente tiempo en la secuencia temporal.

Genera:

`TimeLineInvalidException` – si la secuencia temporal deja de ser válida en el contenido en cuestión, por ejemplo, porque el contenido sea una grabación con desplazamiento temporal y el final de la secuencia temporal ya no está dentro de la memoria intermedia

`TimeOutOfRangeException` – si el tiempo especificado no está dentro de la secuencia temporal.

D.3.8 Interfaz TimeLineControl

```
org.ocap.shared.media
     Interface TimeLineControl
```

Todas las superinterfaces:

javax.media.Control

---

interfaz pública **TimeLineControl**

amplía javax.media.Control

Permite el acceso a las secuencias temporales transmitidas en un contenido.

Resumen del método	
`TimeLine[]`	**`getTimeLines`**`()` Devuelve todas las secuencias temporales transmitidas incluidas en un contenido.

Métodos heredados de la interfaz javax.media.Control
`getControlComponent`

Información detallada del método

getTimeLines

```
public TimeLine[] getTimeLines()
```

Devuelve todas las secuencias temporales transmitidas incluidas en un contenido. Si no existen secuencias temporales transmitidas, se devuelve una matriz de longitud cero.

Devuelve:

Una matriz de secuencias temporales.

D.3.9 Clase TimeLineInvalidException

```
org.ocap.shared.media
     Class TimeLineInvalidException

java.lang.Object
  +--java.lang.Throwable
       +--java.lang.Exception
            +--org.ocap.shared.media.TimeLineInvalidException
```

Todas las interfaces implementadas:

java.io.Serializable

---

clase pública **TimeLineInvalidException**

amplía java.lang.Exception

Esta excepción se devuelve cuando una secuencia temporal deja de ser válida en un contenido para el que se había obtenido. Por ejemplo, cuando el contenido sea una grabación con desplazamiento temporal y el final de la secuencia temporal ya no esté dentro de la memoria intermedia.

Véase también:

Serialized Form

Resumen del constructor

TimeLineInvalidException() Construye una TimeLineInvalidException sin mensaje de información detallada

Métodos heredados de la clase java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Métodos heredados de la clase java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Información detallada del constructor

TimeLineInvalidException

```
public TimeLineInvalidException()
```

Construye una TimeLineInvalidException sin mensaje de información detallada

D.3.10 Clase TimeOutOfRangeException

```
org.ocap.shared.media
     Class TimeOutOfRangeException

java.lang.Object
  +--java.lang.Throwable
       +--java.lang.Exception
            +--org.ocap.shared.media.TimeOutOfRangeException
```

Todas las interfaces implementadas:

java.io.Serializable

---

clase pública **TimeOutOfRangeException**

amplía java.lang.Exception

Esta excepción se devuelve cuando un tiempo o un tiempo de medios está fuera de la gama de valores válidos para una secuencia temporal en concreto.

Véase también:

Serialized Form

Resumen del constructor

TimeOutOfRangeException() Construye una TimeOutOfRangeException sin mensaje de información detallada

Métodos heredados de la clase java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, printStackTrace, toString

Métodos heredados de la clase java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Información detallada del constructor

TimeOutOfRangeException

```
public TimeOutOfRangeException()
```

Construye una TimeOutOfRangeException sin mensaje de información detallada

Bibliografía

[b-DAVIC 1.4.1p9] DAVIC 1.4.1p9, DAVIC 1.4.1 Specification Part 9 – *Information Representation*.

[b-ETSI TS 102 816] ETSI TS 102 816, *Digital Video Broadcasting (DVB); PVR/PDR Extension to the Multimedia Home Platform.*

NOTE – At the time of publication of this Recommendation, the above reference is only available as DVB Bluebook A088.rev1. It will become available from ETSI in due course.

[b-OC-SP-OCAP-DVR-I02-050524] OC-SP-OCAP-DVR-I02-050524, *OpenCable Application Platform Specification; OCAP Digital Video Recorder (DVR)*.

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
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 multimedia
Serie I	Red digital de servicios integrados
Serie J	**Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedia**
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	Gestión de las telecomunicaciones, incluida la RGT y el mantenimiento de redes
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, comunicaciones de sistemas abiertos y seguridad
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, 2008