Unión Internacional de Telecomunicaciones

UIT-T J.280

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

(05/2000)

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

Transmisión digital de señales de televisión

Inserción de programas digitales: Interfaz de programación de aplicaciones de empalme

Recomendación UIT-T J.280

Recomendación UIT-T J.280

Inserción de programas digitales: Interfaz de programación de aplicaciones de empalme

Resumen

En esta Recomendación se define una interfaz de programa de aplicación (API) como método normalizado para la comunicación entre servidores y empalmadores para la inserción de contenido en cualquier múltiplex de salida MPEG-2 del empalmador. Esta API es suficientemente flexible y permite conectar uno o más servidores a uno o más empalmadores. Se consideran como una inserción de programas digitales los contenidos tales como anuncios publicitarios de distinta longitud, programas de sustitución, anuncios del servicio público u otro tipo de programas que se crean empalmando entre porciones del programa procedente del servidor.

Orígenes

Este texto fue aprobado el 14 de diciembre de 2005 como la enmienda 1 a la Recomendación UIT-T J.280 (2004) por la Comisión de Estudio 9 (2005-2008) del UIT-T por el procedimiento de la Recomendación UIT-T A.8. Tras se decidió publicar el texto completo de la Recomendación.

Palabras clave

API, empalmador, empalme, inserción de programas, servidor.

PREFACIO

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

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

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

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

NOTA

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

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

PROPIEDAD INTELECTUAL

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

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

ÍNDICE

Página

Recomendación UIT-T J.180

Inserción de programas digitales: Interfaz de programación de aplicaciones de empalme

1 Alcance

Esta interfaz de programa de aplicación (API, *application program interface*) es un método normalizado para la comunicación entre servidores y empalmadores para la inserción de contenido en cualquier múltiplex de salida MPEG-2 del empalmador. Esta API es suficientemente flexible y permite conectar uno o más servidores a uno o más empalmadores. Se consideran como una inserción de programas digitales los contenidos tales como anuncios publicitarios de distinta longitud, programas de sustitución, anuncios del servicio público u otro tipo de programas que se crean empalmando porciones del programa procedente del servidor.

Esta Recomendación no abarca las instrucciones y el control de la posproducción o la edición (por ejemplo, imágenes superpuestas o efectos de compresión de imagen), ni define cómo realizar un empalme, ni especifica en qué grado el empalme es liso. Además, se considera que el importante tema de la sincronización de los trenes asíncronos que han de empalmarse queda para resolverse en cada implementación, en el dispositivo de empalme [3].

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.

[1] Recomendación UIT-T H.222.0 (2000) | ISO/CEI 13818-1:2000, *Tecnología de la información – Codificación genérica de imágenes en movimiento e información de audio asociada: Sistemas*.

[2] Recomendación UIT-T H.262 (2000) | ISO/CEI 13818-2:2000, *Tecnología de la información – Codificación genérica de imágenes en movimiento e información de audio asociada: Vídeo*.

[3] Recomendación UIT-T J.181 (2004), *Mensaje de aviso de inserción de programa digital para sistemas de televisión por cable*.

2.2 Referencias informativas

[4] Apéndice I a la Recomendación UIT-T J.181 (2004), *Prácticas recomendadas y guía de interpretación*.

[5] IETF RFC 3810 (2004), *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*.

3 Definiciones

En esta Recomendación se definen los términos siguientes.

3.1 conexión API: Una conexión de zócalo TCP/IP entre un servidor y un empalmador para la transferencia de mensajes API.

3.2 inserción adosada: Dos o más sesiones temporalmente contiguas sin retorno al canal primario entre ellas.

3.3 canal: Canal es sinónimo de "Servicio" en la terminología de la difusión de vídeo digital (DVB), y de "Programa" en la terminología MPEG.

3.4 canal de inserción: Canal(es) múltiplex de inserción que sustituye(n) total o parcialmente al canal primario durante el evento de empalme.

3.5 múltiplex de inserción: Es el origen del canal de inserción. Un múltiplex producido por un servidor puede, en determinadas circunstancias, excluir la información específica de programa (PSI, *programme-specific information*), por lo que se entiende que este múltiplex puede ser un tren de transporte MPEG-2 no conforme.

3.6 múltiplex: Un múltiplex es un conjunto de uno o más canales que puede incluir la información de servicio conexa. Un múltiplex es un tren de transporte MPEG-2, con la posible excepción del múltiplex de inserción.

3.7 canal de salida: Es el canal producido en la salida del empalmador.

3.8 múltiplex de salida: Es el tren de transporte MPEG-2 producto de la multiplexación de uno o más canales de salida. El empalmador debe garantizar que la PSI del múltiplex de salida siempre es válida.

3.9 canal primario: Un canal del múltiplex primario que se sustituye en todo o en parte. Un único canal primario puede dar lugar a múltiples canales de salida.

3.10 múltiplex primario: Es el origen de los canales primarios.

3.11 servidor: Dispositivo que proporciona los canales de inserción que han de empalmarse en el canal primario. Este dispositivo se comunica con el empalmador para indicarle cuándo y qué ha de empalmar.

3.12 sesión: Una sesión es la inserción de contenido (como anuncios publicitarios de diversa longitud, programas de sustitución, anuncios del servicio público u otro tipo de programas que se crean empalmando porciones del programa procedente del servidor). Cada sesión está identificada por un único **SessionID** (identificador de sesión).

3.13 inicio de empalme (splice-in): Es el empalme que da comienzo a la inserción, en el momento especificado en el mensaje **Splice_Request** (petición de empalme).

3.14 fin de empalme (splice-out): Es el empalme que pone fin a la inserción. El final de la inserción se obtiene sumando el tiempo de comienzo y la duración especificada en el mensaje **Splice_Request**, pero la inserción podría terminar antes si hay un estado de error.

3.15 empalmador: Es el dispositivo que empalma el/los canal(es) de inserción en el/los canal(es) primario(s). Puede recibir mensajes de aviso J.181. Este dispositivo se comunica con el servidor para saber cuándo y qué ha de empalmar.

4 Abreviaturas, siglas o acrónimos

En esta Recomendación se utilizan las siguientes abreviaturas, siglas o acrónimos.

API Interfaz de programa de aplicación (*application program interface*)

CNN Cable News Network

DVB-ASI Interfaz serie asíncrona de difusión de vídeo digital (*digital video broadcast – asynchronous serial interface*)

ID Identificador (*identifier*)

ISO Organización Internacional de Normalización (*International Organization for Standardization*)

MLD Descubrimiento de oyente multidifusión (*multicast listener discovery*)

MPEG Grupo de expertos en imágenes en movimiento (*moving picture experts group*)

MPTS Tren de transporte multiprograma (*multi-program transport stream*)

NCTA National Cable Television Association

NTP Protocolo de señales horarias de red (*network time protocol*)

PAT Tabla de asociación de programas (*program association table*)

PCR Referencia de reloj de programa (*program clock reference*)

PID Identificador de paquete (*packet identifier*)

PMT Tabla de correspondencia de programa (*program map table*)

PSI Información específica de programa (*program specific information*)

SCTE Asociación de Ingenieros de Telecomunicaciones por Cable (*Society of Cable Telecommunications Engineers*)

SPTS Flujo de transporte de programa único (*single program transport stream*)

TCP/IP Protocolo de control de transporte/protocolo Internet (*transport control protocol/ Internet protocol*)

uimsbf Entero sin signo, bit más significativo primero (*unsigned integer, most significant bit first*)

UIT Unión Internacional de Telecomunicaciones

UTC Tiempo universal coordinado (*coordinated universal time*)

5 Notación de conformidad

En la presente Recomendación, la utilización del futuro indica la obligatoriedad de la norma. La palabra "*debe*" indica una disposición recomendada pero no obligatoria. La palabra "*puede*" o "*podrá*" implica una característica cuya presencia no excluye la conformidad y que puede o no estar presente, a voluntad del implementador.

6 Introducción

6.1 Diagrama sinóptico del sistema

Esta API puede utilizarse con diferentes configuraciones de servidor(es) y empalmador(es). La presente interfaz de programa de aplicaciones se basa en una configuración de un único servidor y un único empalmador, como se muestra en la figura 1. No obstante, esta configuración puede ampliarse a múltiples servidores y múltiples empalmadores, como se muestra en la figura 2.

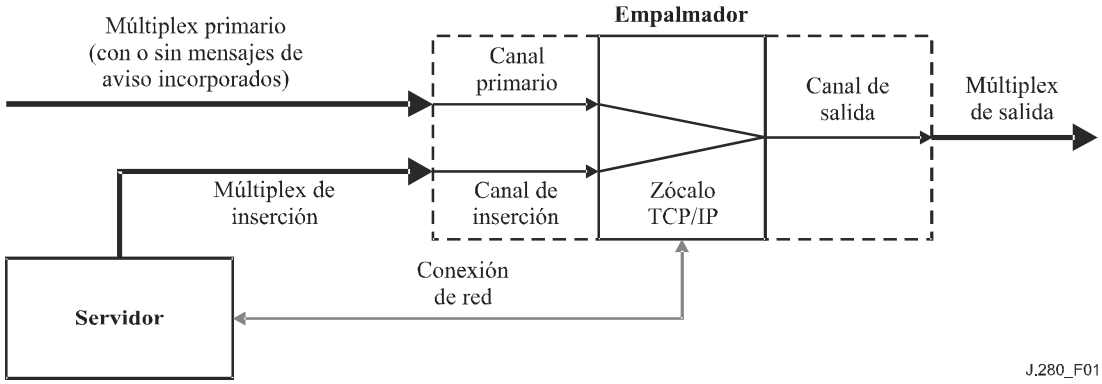


Figura 1/J.280 – Un servidor/un empalmador

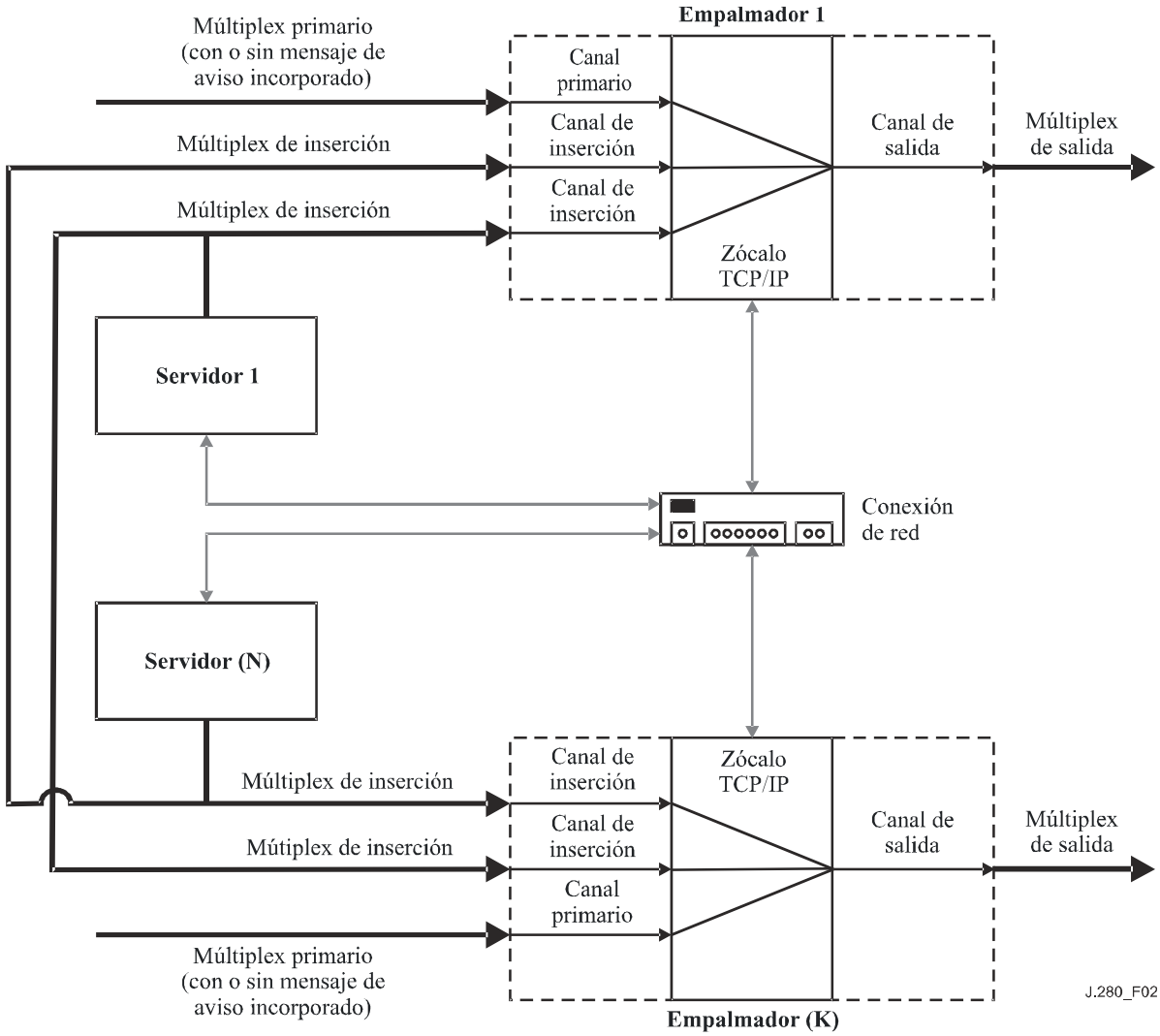


Figura 2/J.280 – Múltiples servidores/múltiples empalmadores

El modelo en esta API tiene un empalmador con una o más entradas de múltiplex. El empalmador separa lógicamente los canales en los múltiplex y los presenta a un conmutador, que puede establecer la correspondencia entre cualquier entrada y cualquier canal de salida. En la

configuración inicial hay una correspondencia entre los canales primarios y los canales de salida, pero el servidor puede indicar al empalmador que pase de un canal primario a un canal de inserción durante un periodo específico. Seguidamente, puede indicar al empalmador que pase a otro canal de inserción después del primer cambio.

Lógicamente, un empalme conlleva la existencia de dos canales de entrada y un canal de salida. El empalmador es el dispositivo que une los diversos trenes elementales (audio, vídeo y datos). El punto de empalme óptimo puede darse en momentos ligeramente diferentes para cada tren elemental, por lo que el empalmador debe realizar el empalme que proporcione la mejor calidad de salida. El empalme no siempre se realiza a partir de la "red de programación" del canal primario hacia el "anuncio" del canal de inserción y de vuelta a la "red de programación" del canal primario. El empalmador puede empalmar contenido que esté almacenado únicamente en el servidor y que llega a través de un único múltiplex de entrada. Es posible utilizar esta API para una configuración diferente: el servidor tiene una salida de tren de transporte multiprograma (MPTS, *multiprogram transport stream*), que contiene programa y material intermedio, utiliza el empalmador para crear los empalmes adecuados entre los elementos de contenido.

Esta API soporta todas las combinaciones entre un único o múltiples servidores que comuniquen con un único o múltiples empalmadores. Hay una conexión API asociada con cada canal de salida.

En algunas configuraciones puede haber múltiples servidores o múltiples canales dentro del múltiplex de inserción conectado a un empalmador. En estos casos, el empalmador tendrá múltiples conexiones API asociadas con un canal de salida. Cuando se recibe un mensaje de aviso J.181 en un canal primario, el mensaje **Cue_Request (petición de aviso)** debe enviarse a los servidores a través de todas las conexiones API creadas para los canales de salida correspondientes. También es posible que más de una conexión API transporte el mensaje **Splice_Request** para la misma inserción en el mismo momento en un canal de salida.

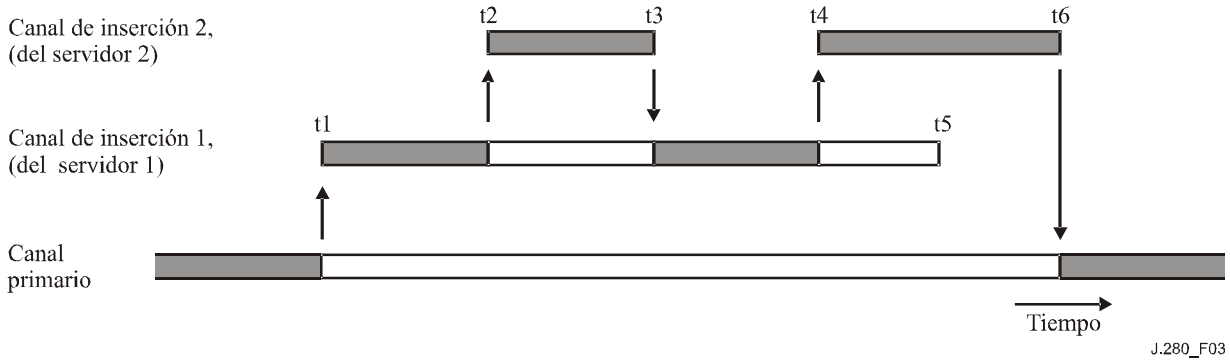
6.2 Prioridades de arbitraje

Se utilizan diversos niveles de acceso para garantizar que se empalma el canal de inserción correcto. Hay diez niveles de acceso distintos, de 0 a 9. El nivel 9 representa la mayor prioridad y puede anular cualquier conexión de menor prioridad. La bandera **OverridePlaying (anular reproducción)** en el mensaje **Splice_Request** especifica si una petición de inserción se acepta cuando el empalmador ya está poniendo en cola otras inserciones o realizándolas. Si la bandera se pone a 1, la inserción de mayor prioridad puede hacer que se interrumpan las inserciones que se estén realizando en ese momento con igual o menor prioridad. Si la bandera se pone a 0, el empalmador no sustituirá la inserción que se esté realizando, incluso si la nueva petición tiene mayor prioridad.

El mensaje **Splice_Request** debe enviarse al menos tres segundos antes del momento de empalme para que sea válido. Si no se cumple el requisito mínimo de tres segundos, el resultado del mensaje **Splice_Request** no queda determinado por la presente API. Si múltiples servidores inician peticiones de empalme para el mismo momento y con la misma prioridad, el empalmador decidirá qué petición aceptar aplicando el criterio del orden de llegada. Todas las demás peticiones se denegarán y se enviará un error de colisión en el mensaje **Splice_Response (respuesta de empalme)** (a menos que esté puesta a 1 la bandera **OverridePlaying**).

Durante el periodo inmediatamente anterior al inicio de una inserción, si se recibe una **Splice_Request** de prioridad 5 para el mismo momento de empalme que una **Splice_Request** de prioridad 3, se devuelve un error de colisión relativo a la petición de prioridad 3. Si se recibe posteriormente una **Splice_Request** de prioridad 7, se enviará el error de colisión a la petición de prioridad 5 y se pondrá a la cola la petición de prioridad 7. Si se recibe una segunda prioridad 7 con la bandera **OverridePlaying** puesta a 0, se notificará error de colisión para esta segunda petición de prioridad 7. No obstante, si la bandera **OverridePlaying** está puesta a 1 en la segunda petición de prioridad 7, la primera petición de prioridad 7 será el objeto del error de colisión y quedará anulada.

En la figura 3 se muestran tres entradas de un empalmador. Las zonas sombreadas indican qué entrada se dirigirá al canal de salida en un momento dado.



t1 – El servidor 1 envía un mensaje **Splice_Request** y envía su tren al empalmador. El empalmador conecta este tren del canal de inserción al canal de salida. En el mensaje **Splice_Request** se solicita una duración de inserción desde el momento t1 hasta el momento t5. El empalmador enviará al servidor 1 un mensaje **SpliceComplete_Response** (**respuesta de empalme terminado**) con la bandera **SpliceType** (**tipo de empalme**) puesta a Splice_in (inicio de empalme) y un código de resultado 100 "Respuesta de éxito".

t2 – El servidor 2 envía un mensaje **Splice_Request** con la bandera **OverridePlaying** puesta a 1, y hay una solicitud con la misma prioridad superior. En el momento especificado por la **Splice_Request**, el tren del canal de inserción 2 se conecta al canal de salida (sustituyendo el tren antes conectado procedente del servidor 1). En **Splice_Request** del servidor 2 se solicita una duración desde el momento t2 al momento t3. El empalmador enviará al servidor 1 un mensaje **SpliceComplete_Response** con la bandera **SpliceType** puesta a Splice_out (fin de empalme) y un código de resultado 125 "Anulación del canal". El empalmador enviará al servidor 2 un mensaje **SpliceComplete_Response** con la bandera **SpliceType** puesta a Splice_in y un código de resultado 100 "Respuesta de éxito". Si el servidor 1 determina que la anulación del canal es un error, puede enviar una solicitud **Abort_Request** (**cancelar petición**) y dar por terminado su tren en ese momento. Esta posibilidad no se muestra en la figura 3.

t3 – La duración de inserción se ha completado y el empalmador vuelve al material procedente del servidor 1 y destinado al canal de salida. Cabe señalar que el empalmador no vuelve al canal primario para dirigirlo al canal de salida. El empalmador enviará al servidor 1 un mensaje **SpliceComplete_Response** con la bandera **SpliceType** puesta a Splice_in y un código de resultado 125 "Anulación del canal". El empalmador enviará al servidor 2 un mensaje **SpliceComplete_Response** con la bandera **SpliceType** puesta a Splice_out y un código de resultado 100 "Respuesta de éxito".

t4 – El servidor 2 envía otra solicitud **Splice_Request** con la bandera **OverridePlaying** puesta a 1. En el momento especificado por **Splice_Request**, el tren del canal de inserción 2 se conecta al canal de salida (sustituyendo el tren conectado hasta el momento procedente del servidor 1). En **Splice_Request** del servidor 2 se solicita una duración desde el momento t4 al momento t6. El empalmador enviará al servidor 1 un mensaje **SpliceComplete_Response** con la bandera **SpliceType** puesta a Splice_out y un código de resultado 125 "Anulación del canal". El empalmador enviará al servidor 2 un mensaje **SpliceComplete_Response** con la bandera **SpliceType** puesta a Splice_in y un código de resultado 100 "Respuesta de éxito".

t5 – El tren de inserción del servidor 1 termina con 2 porciones de su duración reproducidas y 2 porciones anuladas por el tren del servidor 2.

t6 – Se ha completado la duración de inserción y el empalmador vuelve al material procedente del canal primario para conectarlo al canal de salida. El empalmador enviará al servidor 2 un mensaje **SpliceComplete_Response** con la bandera **SpliceType** puesta a Splice_out y un código de resultado 100 "Respuesta de éxito".

Figura 3/J.280 – Funcionamiento de la bandera OverridePlaying (anular reproducción)

También es posible que varios servidores intenten dividir un mensaje **Cue_Request**. Puede citarse como ejemplo una oportunidad de empalme de 60 segundos de duración durante la cual un servidor utilizará los primeros 30 segundos y el segundo los últimos 30. Dependiendo de las prioridades y de cuándo se reciban los mensajes **Splice_Request** el empalmador podría indicar un código de resultado 109 (Colisión de empalme). Esta API no coordina la capacidad de dos servidores para realizar esta funcionalidad, lo que puede hacerse mediante acuerdo mutuo entre servidores o utilizando una API servidor a servidor.

6.3 Terminaciones anormales

Es posible que una inserción se anule en algún momento durante la reproducción debido a una inserción de mayor prioridad. En este caso, el empalmador volverá a la inserción anulada al final de la inserción de mayor prioridad. Si la inserción de mayor prioridad fuera anulada por medio de un

mensaje **Abort_Request**, el empalmador volvería a la primera inserción anulada. Si el canal de inserción inicial ya no está disponible, el empalmador volverá al canal primario, si es posible.

Si el servidor solicita un empalme en un canal primario que en ese momento no tiene entrada válida, el empalmador realizará el empalme, pero enviará un mensaje **SpliceComplete_Response** al servidor con código de resultado 111 (Canal primario no encontrado). Del mismo modo, la acción de empalme de un canal de inserción hacia un canal primario sin entrada válida producirá un código de resultado 111 (Canal primario no encontrado).

Podría ser conveniente instalar soporte lógico en el empalmador para garantizar que se vuelve siempre al canal primario. Es muy aconsejable asegurarse de que el empalmador volverá al canal primario cuando se dé una condición de error que interrumpa la transmisión del canal de salida. En este caso, el empalmador devolverá el mensaje **SpliceComplete_Response** con un código de resultado 110 (Canal de inserción no encontrado).

6.4 Requisitos de empalme

El empalmador necesita información sobre el canal de inserción antes de que pueda empalmarlo con el canal primario. Parte de esta información se enviará a través de la conexión API y otra parte a través del múltiplex MPEG. Es necesario tener toda la información antes de realizar el empalme.

ChannelName (nombre del canal) es la identificación del canal de salida. Se trata de un nombre único asignado a cada canal de salida (por ejemplo, CNN) en la configuración del empalmador, que el servidor necesita para determinar qué canal primario será sustituido por cada canal de inserción.

El empalmador necesita conocer el canal de inserción a empalmar en el canal primario: la ubicación del múltiplex de inserción y el canal del múltiplex de inserción a utilizar. Esta información está disponible en el mensaje **Splice_Request**.

6.5 Comunicación

La comunicación entre el servidor y el empalmador se realiza a través de una conexión de zócalo TCP/IP para cada canal de salida. Un zócalo TCP/IP se define por la dirección IP y el número de puerto TCP del servidor y del empalmador. Las direcciones IP son únicas para cada servidor o empalmador y los números de puerto TCP son únicos para cada programa que ha de empalmarse. Así se identifican unívocamente el servidor de anuncios, el empalmador y la conexión aplicable a cada canal de salida. Una vez establecida, esta conexión API permanece activa hasta que uno de los dispositivos la dé por terminada; después es necesario hacer una reinicialización para volver a empalmar.

Todos los mensajes intercambiados entre el empalmador y el servidor comparten un formato general común que se detalla en 7.1. Todos los mensajes para la comunicación entre el empalmador y el servidor deben ajustarse a este formato. El formato permite mensajes del tipo "Definidos por el usuario" que pueden utilizarse como modelo para los mensajes de datos privados entre servidores y empalmadores que no se consideran en esta Recomendación.

Todos los mensajes de petición requieren una respuesta del empalmador o del servidor, dependiendo del origen de la petición. La mayor parte de los mensajes de respuesta sólo indican un resultado y no contienen más datos. Estos mensajes son necesarios para asegurar al emisor de la petición que se ha recibido e interpretado correctamente el mensaje. De haber errores, el mensaje puede ser reenviado.

6.6 Temas que quedan en estudio

Esta Recomendación especifica un periodo de planificación de empalme mínimo de tres segundos entre la recepción de la petición de empalme y el empalme efectivo. La definición, los parámetros y la aplicación de una "petición de empalme inmediato" con un periodo de espera sustancialmente inferior será tema de estudios posteriores.

7 Sintaxis de la API

7.1 Sintaxis de Splicing_API_Message (mensaje API de empalme)

Todos los mensajes en esta API contienen una estructura general que envuelve los datos del mensaje específico que se envía. Así, al recibir el mensaje, una rutina de análisis sintáctico puede almacenar el mensaje, determinar la estructura de los datos y garantizar que el mensaje se recibe correctamente.

Cuadro 7-1/J.280 – Splicing_API_Message (Mensaje API de empalme)

Sintaxis	Bytes	Tipo
Splicing_API_Message {		
MessageID	2	uimsbf
MessageSize	2	uimsbf
Result	2	uimsbf
Result_Extension	2	uimsbf
data( )	*	*
}		

MessageID (identificador de mensaje) – Valor entero que indica qué mensaje se envía. Véase el cuadro 7-2.

MessageSize (tamaño de mensaje) – Tamaño del campo data () (datos) que se envía, en bytes.

Result (resultado) – El resultado en el mensaje de respuesta. Véanse en el apéndice I los detalles de los códigos de resultado. En los mensajes de petición, se pone a 0xFFFF.

Result_Extension (extensión de resultado) – Debe estar puesto a 0xFFFF, a menos que se utilice para enviar una información de resultado adicional en un mensaje de respuesta.

data( ) (datos) – Estructura de datos específica para el mensaje que se envía. A continuación se describen los distintos mensajes que contienen datos. El tamaño de este campo es igual al de **MessageSize** y queda determinado por el tamaño de los datos que se añaden al mensaje. No todos los mensajes utilizan el campo data( ).

Cuadro 7-2/J.280 – Valores de MessageID

MessageID	Nombre del mensaje	Enviado por	Descripción
0x0000	General_Response (respuesta general)	Empalmador o servidor	Utilizado para transportar información asíncrona entre los dispositivos. No hay campo data( ) asociado con este mensaje.
0x0001	Init_Request (petición inicial)	Servidor	Mensaje inicial al empalmador en el puerto 5168.
0x0002	Init_Response (respuesta inicial)	Empalmador	Respuesta inicial al servidor a través de la conexión establecida.
0x0003	ExtendedData_Request (petición de datos ampliada)	Servidor	Petición al empalmador para conocer información de reproducción detallada.
0x0004	ExtendedData_Response (respuesta de datos ampliada)	Empalmador	Respuesta propia del proveedor con datos de reproducción más completos para el evento solicitado.
0x0005	Alive_Request (petición de actividad)	Servidor	Envía un mensaje de actividad para conocer el estado actual.
0x0006	Alive_Response (respuesta de actividad)	Empalmador	Respuesta al mensaje de actividad indicando el estado actual.
0x0007	Splice_Request (petición de empalme)	Servidor	Solicitud de empalme en un momento específico.
0x0008	Splice_Response (respuesta de empalme)	Empalmador	Respuesta para indicar que el empalmador ha recibido **Splice_Request** y está listo para empalmar.
0x0009	SpliceComplete_Response (respuesta empalme terminado)	Empalmador	Respuesta en splice-in y splice-out.
0x000A	GetConfig_Request (petición de configuración)	Servidor	Solicitud para conocer la configuración de empalme actual para esta conexión API.
0x000B	GetConfig_Response (respuesta de configuración)	Empalmador	Contiene toda la información sobre el empalme de la conexión API.
0x000C	Cue_Request (petición de aviso)	Empalmador	El empalmador envía una sección de información de aviso al servidor.
0x000D	Cue_Response (respuesta de aviso)	Servidor	Acuse de recibo de la sección de información de aviso.
0x000E	Abort_Request (petición de cancelar)	Servidor	Solicitud de volver inmediatamente al canal primario o al canal de inserción anulado.
0x000F	Abort_Response (respuesta de cancelar)	Empalmador	Acuse de recibo del mensaje **Abort_Request**. También se generará, de ser necesario, un mensaje SpliceComplete_Response.
0x0010 - 0x7FFF 0xFFFF	Reservado		Valores reservados para la futura normalización.
0x8000 - 0xFFFE	Definido por el usuario		Valores disponibles para las funciones definidas por el usuario.

7.2 Convenios y requisitos

1) A continuación se enumeran mensajes que contienen datos, con sus correspondientes campos y tipos de datos. Las estructuras adicionales se indican como funciones y se describen en la cláusula 8.

2) La longitud de todas las cadenas de caracteres tiene un espacio reservado para un carácter de terminación nulo, y siempre se utilizan cadenas terminadas con un valor nulo. Por ejemplo, una cadena de 16 caracteres sólo podrá contener como máximo 15 caracteres de datos seguidos de un carácter nulo (0x00) inmediatamente después del último carácter de datos. Una vez que se encuentra el valor nulo al examinar la cadena, el resto de caracteres en la cadena queda sin definir. El tamaño definido para la cadena de caracteres es constante y no variará dependiendo de la longitud de la cadena. La presente Recomendación utiliza cadenas con caracteres ASCII de 8 bits.

3) Todos los valores temporales se expresan en tiempo universal coordinado (UTC, *coordinated universal time*).

4) En esta Recomendación se pone todo a 1 para indicar que la opción es CUALQUIERA. Para un campo de 4 bytes este valor será 0xFFFFFFFF.

5) Los mensajes de respuesta se enviarán sin retardo innecesario. El dispositivo que espera considerará que no hay respuesta y que ha expirado la temporización si no recibe nada en el plazo de 5 segundos. Cuando un servidor suponga que ha expirado la temporización, enviará un mensaje Alive_Request. Si el empalmador no responde como se especifica en la presente Recomendación, se abandonará la conexión para dicho canal y volverá a establecerse.

6) Un servidor que reciba un mensaje de respuesta en el que se indica la incapacidad de analizar sintácticamente un mensaje (código de error 123) enviará un mensaje Alive_Request. Si no recibe el mensaje Alive_Response adecuado, se abandonará la conexión para este canal y volverá a establecerse.

7) Se utiliza el campo Result en el Splicing_API_Message para devolver un código de resultado. Pueden devolverse múltiples códigos de respuesta enviando varios mensajes **General_Response** en cualquier momento.

8) Si el empalmador o el servidor no pueden analizar sintácticamente el mensaje de petición, devolverán el mensaje **General_Response** con un código de resultado 123.

7.3 Inicialización

La comunicación inicial comienza cuando el empalmador está a la espera en el puerto 5168 predefinido y un servidor abre una conexión API hacia el empalmador. El servidor envía un mensaje **Init_Request** al empalmador, y espera la respuesta del empalmador a través de la conexión API establecida. Cualquier comunicación posterior se realizará a través de esta conexión API. El empalmador o el servidor pueden dar por terminada la comunicación cerrando esta conexión. Cada dispositivo es responsable de detectar y tratar adecuadamente una conexión API cerrada. Cuando el empalmador inicializa el TCP para la espera en el puerto 5168, debe prever como mínimo el triple del número de canales de inserción para las conexiones API hacia el empalmador. Por ejemplo, si el empalmador controla 70 canales, de los cuales 40 son empalmables, debe prever 120 (40 × 3), conexiones API simultáneas.

7.3.1 Mensaje Init_Request (petición inicial)

El campo data( ) de este mensaje contiene la estructura Init_Request_Data que se señala en el cuadro 7-3.

Cuadro 7-3/J.280 – Init_Request_Data

Sintaxis	Bytes	Tipo
Init_Request_Data {		
Version( )		
ChannelName	32	Cadena de caracteres
SplicerName	32	Cadena de caracteres
Hardware_Config( )		
for (i=0; i<N; i++)		
splice_API_descriptor( )		
}		

Version( ) (versión) – Véase 8.1.

ChannelName (nombre del canal) – Nombre lógico que se da al canal de salida de esta conexión. También se utiliza para verificar la conexión API correcta cuando el empalmador responde al servidor.

SplicerName (nombre del empalmador) – Nombre del dispositivo de empalme, si el servidor utiliza la API para comunicar con un dispositivo que controla múltiples empalmadores.

Hardware_Config( ) (configuración de soporte físico) – Véase 8.2.

splice_API_descriptor( ) (descriptor API de empalme) – Descriptor que debe seguir a la sintaxis definida en 8.5. El descriptor missing_Primary_Channel_action_descriptor( ) (falta descriptor de acción de canal primario) es adecuado para esta petición.

7.3.2 Mensaje Init_Response (respuesta inicial)

El empalmador responde a un **Init_Request** enviando un mensaje **Init_Response** a través de la conexión API abierta. El servidor comprueba si soporta la versión enviada por el empalmador y si dispone de una conexión API establecida hacia el canal primario correcto.

El campo data( ) de este mensaje contiene la estructura Init_Response_Data que se presenta en el cuadro 7-4.

Cuadro 7-4/J.280 – Init_Response_Data

Sintaxis	Bytes	Tipo
Init_Response_Data {		
Version( )		
ChannelName	32	Cadena de caracteres
}		

Version( ) – Véase 8.1. El empalmador responderá indicando el número de versión de la API más alto que es capaz de soportar.

ChannelName – Se devuelve al servidor para indicar que se ha realizado la conexión correcta.

7.4 Mensajes de aviso incorporados

Algunos empalmadores pueden recibir mensajes de aviso basados en la Rec. UIT-T J.181, que se han de remitir al servidor. Se utiliza el mensaje **Cue_Request (petición de aviso)** para transmitir estos mensajes de aviso del empalmador al servidor. Cuando un empalmador recibe un mensaje de aviso, envía al servidor toda la información splice_info_section( ) (sección de información de empalme) junto con el tiempo de empalme. El servidor acusará recibo del mensaje respondiendo

con un mensaje **Cue_Response (respuesta de aviso)**. El mensaje **Cue_Response** consiste únicamente en un Splicing_API_Message (mensaje API de empalme) sin campo data( ) asociado, pero puede incluir un código de retorno. Si está criptado, el empalmador descriptará splice_info_section( ) antes de enviarlo al servidor.

Si el empalmador recibe un mensaje de aviso y detecta que está alterado, enviará un **General_Message** al servidor con el código de resultado 117 (Mensaje de aviso no válido). El empalmador no enviará ningún mensaje **Cue_Request** en este caso.

7.4.1 Mensaje Cue_Request (petición de aviso)

El campo data( ) de este mensaje contiene la estructura Cue_Request_Data que se presenta en el cuadro 7-5.

Cuadro 7-5/J.280 – Cue_Request_Data

Sintaxis	Bytes	Tipo

Cue_Request_Data {
time( )
splice_info_section( )
}		

time( ) (tiempo) – El empalmador deduce este tiempo del splice_time( ) (tiempo de empalme) indicado en splice_info_section( ) del mensaje de aviso J.181. Si en splice_info_section J.181 se ha indicado el modo empalme de componente, time( ) se referirá al tiempo de empalme por defecto que se indica en 7.5.2.1/J.181. En caso de que splice_info_section( ) no contenga un pts_time( ) (tiempo de presentación) que sea necesario traducir, como en la instrucción splice_schedule( ) (plan de empalme), la estructura de tiempo se completará toda con 1s para denotar que no se especifica el tiempo. Corresponde al empalmador determinar cómo convertir el tiempo PTS en tiempo universal (UTC) para comunicar con el servidor. Este proceso puede variar según los empalmadores, dependiendo de cómo gestionen adecuadamente sus memorias intermedias internas. Véase la sintaxis de la estructura time( ) en 8.4.

splice_info_section( ) – Pueden encontrarse los detalles de esta estructura en la Rec. UIT-T J.181.

7.5 Mensajes de empalme

Una vez inicializado y configurado el empalmador, el servidor puede enviar un mensaje **Splice_Request** para iniciar una sesión. Los dos mensajes que se devuelven a partir del mensaje **Splice_Request** son **Splice_Response** y **SpliceComplete_Response (respuesta de empalme terminado)**. El servidor enviará un mensaje **Splice_Request** al menos 3 segundos antes del instante time( ) indicado en el mensaje **Splice_Request**. Esto permite al empalmador actualizar su configuración y prepararse para el empalme. El tren del canal de inserción para dicha sesión debe comenzar entre 300 y 600 milisegundos antes del instante time( ) medido a la entrada del empalmador. Debe enviarse una referencia de reloj de programas (PCR, *program clock reference*) en la primera unidad de acceso de vídeo del tren de canal de inserción, o antes de ella. El tren de vídeo del contenido de inserción comenzará con una cabecera de secuencia y una trama I. El empalmador podrá albergar al menos 10 mensajes **Splice_Request** en cola para una conexión API. Si la cola de mensajes del empalmador está llena, éste responderá con un código de resultado 114 (Cola de empalme llena).

Los detalles de la conexión física se proporcionan en el mensaje **Init_Request**. Hay dos maneras de indicar cuáles son el canal en el múltiplex de inserción y qué identificador de paquete (PID, *paquet identifier*) que se van a utilizar:

- Si **ServiceID (identificador de servicio)** no es 0xFFFF en el mensaje **Splice_Request**, el campo ServiceID especifica un número de la tabla de asociación de programas (PAT, *program association table*), al que corresponde una tabla de correspondencia de programa (PMT, *program map table*). La PAT y la PMT deben ser estables en el canal de inserción al menos 200 ms antes de enviar el mensaje **Splice_Request** y deben mantenerse estables durante toda la sesión. Debe tratarse de tablas MPEG normalizadas actualizadas, según convenga.
- Si **ServiceID** es 0xFFFF, se utiliza la estructura splice_elementary_stream( ) (empalme de tren elemental) (PCR, vídeo, audio y PID de datos) en el mensaje **Splice_Request**.

NOTA – Si se utiliza este método, el ServiceID se pondrá a 0xFFFF. El empalmador proporcionará al múltiplex de salida un tren de transporte MPEG-2 conforme, aunque el múltiplex de inserción no necesita incluir la PSI.

Es importante el orden en que se envían los mensajes de empalme. El primer mensaje enviado para una determinada secuencia de inserciones adosadas estará determinado por el valor time( ), mientras que todos los demás mensajes **Splice_Request** pueden utilizar **PriorSession (sesión anterior)**. El número **PriorSession** debe hacer referencia a una sesión existente que aún no se ha completado. En cualquier otro caso, se devuelve un error 123 señalando el **PriorSession** o el campo time( ).

El servidor elige los PID de los trenes elementales dentro de un múltiplex de inserción. Los PID no pueden ser comunes a dos sesiones adyacentes procedentes del mismo servidor y enviadas a través del mismo múltiplex de inserción, puesto que los trenes de sesiones adyacentes en ocasiones pueden solaparse ligeramente en el tiempo por requisitos de esta API.

7.5.1 Mensaje Splice_Request (petición de empalme)

El campo data( ) de este mensaje contiene la estructura Splice_Request_Data que se presenta en el cuadro 7-6.

Cuadro 7-6/J.280 – Splice_Request_Data

Sintaxis	Bytes	Tipo
Splice_Request_Data {		
SessionID	4	uimsbf
PriorSession	4	uimsbf
time( )		
ServiceID	2	uimsbf
If (**ServiceID** = 0xFFFF)		
{		
PcrPID	2	uimsbf
PIDCount	4	uimsbf
for (j=0; j<**PidCount**; j++)		
splice_elementary_stream( )		
}		
Duration	4	uimsbf
SpliceEventID	4	uimsbf
PostBlack	4	uimsbf
AccessType	1	uimsbf
OverridePlaying	1	uimsbf
ReturnToPriorChannel	1	uimsbf
for (i=0; i<N; i++)		
splice_API_descriptor( )		
}		

SessionID – Identificador de la sesión. Se utiliza para distinguir esta petición de otras peticiones que estén expidiéndose o vayan a expedirse. No se permite que haya múltiples mensajes **Splice_Request** concurrentes con el mismo **SessionID**. Si se utiliza **ExtendedData_Request (petición de datos ampliados)**, deberá recibirse una **ExtendedData_Response (respuesta de datos ampliados)** para el **SessionID** antes de que éste pueda volver a utilizarse.

PriorSession – Este campo facilita las inserciones adosadas. El valor de este campo contendrá el **SessionID** de la sesión inmediatamente anterior. Si es 0xFFFFFFFF, esta sesión utiliza el valor time( ) para iniciar su inserción, en vez del **SessionID** de la sesión anterior. Este campo tendrá un **SessionID** válido únicamente cuando la sesión inmediatamente anterior proceda del mismo servidor. El campo time( ) debe utilizarse en vez del campo **PriorSession** cuando se creen inserciones adosadas procedentes de diversos servidores.

time( ) – Es el instante de empalme del evento. Este campo será normalmente el campo time( ) del mensaje **Cue_Request** que se reenvía al empalmador. En eventos no iniciados mediante **Cue_Request**, se tratará del momento en que el servidor pretenda forzar un evento de empalme. No se tendrá en cuenta este campo si **PriorSession** no es igual a 0xFFFFFFFF. Si este valor no está relacionado con un mensaje de aviso J.181, puede variar de un empalmador a otro el momento en que ocurre realmente el empalme, dependiendo de su memoria intermedia y del tipo de empalme. Véase en 8.4 la sintaxis de la estructura de time( ).

ServiceID – Es el número de programa del canal en el múltiplex de inserción que se empalmará en lugar del canal primario. Si está puesto a 0xFFFF, se necesita conocer el splice_elementary_stream( ) y el **PIDCount (total de PID)**.

PCR (referencia de reloj de programa) – Indica el PID de la PCR.

PIDCount (total de PID) – Es el número de PID en el canal de inserción (excluido el PID de la PCR).

Duration (duración) – Es el número de tics del reloj a 90 kHz que el servidor pide al empalmador que inserte. Este campo puede anular el valor de duración J.181. Puede ponerse a 0 para indicar al empalmador que conecte al canal de inserción hasta que llegue una nueva **Splice_Request**.

SpliceEventID (identificador de evento de empalme) – Se utiliza para relacionar este evento de inserción con el mensaje de aviso J.181 que puede haber originado el empalme. Debe ser equivalente al splice_event_id (identificador de evento de empalme) de la instrucción splice_insert (instertar empalme) del correspondiente mensaje de aviso J.181. Este valor debe ser idéntico en todos los mensajes **Splice_Request** que pertenezcan al mismo mensaje de aviso J.181. En el caso de un evento no iniciado por un mensaje de aviso J.181, este campo se pondrá a 0xFFFFFFFF.

PostBlack (negro posterior) – Es el número de tics del reloj a 90 kHz de vídeo en negro y audio mudo que han de reproducirse después de reproducir el contenido de inserción. El intervalo **PostBlack** va después de **Duration** y no se incluye en ese tiempo. Si no se solicita ningún **PostBlack**, este campo se pondrá a 0. A los efectos de la bandera **OverridePlaying**, no se considera que **PostBlack** es parte de la inserción que se reproduce.

AccessType (tipo de acceso) – Indica el tipo de acceso que tiene la conexión. Se trata de un entero de 0 a 9, representando 0 la menor prioridad y 9 la mayor.

OverridePlaying – Cuando esta bandera es 0, esta petición **Splice_Request** no puede anular una inserción que se está reproduciendo. Si esta bandera se pone a 1, esta petición **Splice_Request** anulará cualquier inserción de prioridad igual o inferior que se reproduzca en ese momento. Se considera como inserción que se está reproduciendo la que ocurre entre los puntos splice-in y splice-out.

ReturnToPriorChannel (volver al canal anterior) – Cuando esta bandera es 0, el empalmador no volverá al canal primario ni al canal de inserción anulado una vez ejecutada la petición **Splice_Request**. Se espera la emisión de otra petición **Splice_Request** antes de que se complete la inserción. Si no se recibe una nueva **Splice_Request**, el empalmador dejará de transmitir a través de este canal de salida. Si esta bandera se pone a 1, el empalmador volverá al canal anterior a menos que reciba una **Splice_Request** que indique lo contrario.

splice_API_descriptor( ) (descriptor API de empalme) – Es un descriptor que debe seguir a la sintaxis que se define en 8.5. playback_descriptor( ) (descriptor de reproducción) y muxpriority_descriptor( ) (descriptor de prioridad del múltiplex) son descriptores adecuados para esta cláusula.

7.5.2 Mensaje Splice_Response (respuesta de empalme)

El mensaje **Splice_Response** no contiene datos y acusa la recepción del mensaje **Splice_Request**. Este mensaje puede contener un código de error, de ser necesario.

7.5.3 Mensaje SpliceComplete_Response (respuesta de empalme terminado)

El mensaje **SpliceComplete_Response** se envía al inicio y al final de una inserción, también para las inserciones adosadas. Por ejemplo, cuando se reproducen dos contenidos, se devuelven cuatro mensajes **SpliceComplete_Response**, uno al principio del primer contenido, uno cuando se ha completado la inserción, uno al principio del segundo contenido y uno cuando se ha completado la inserción. El código de resultado en la cabecera indicará adecuadamente el motivo de fallo, si no se realiza el empalme, para que el servidor pueda tomar las medidas adecuadas. El splice-in y splice-out son eventos distintos que deben tratarse como tales. Si no se realiza el empalme entre dos contenidos, el evento de salida (*splice-out*) indicará que el estado es normal, si el contenido vigente se ha reproducido íntegramente. El mensaje **SpliceComplete_Response** se enviará inmediatamente

en caso de fallo de cualquier evento de empalme, sin esperar a que termine la duración prevista del contenido insertado.

El campo data( ) de este mensaje contiene la estructura SpliceComplete_Response_Data que se indica en el cuadro 7-7.

Cuadro 7-7/J.280 – SpliceComplete_Response_Data

Sintaxis	Bytes	Tipo
SpliceComplete_Response_Data {		
SessionID	4	uimsbf
SpliceTypeFlag	1	uimsbf
Bitrate	4	uimsbf
PlayedDuration	4	uimsbf
}		

SessionID – Es el SessionID que el mensaje **Splice_Request** ha utilizado.

SpliceTypeFlag (bandera de tipo de empalme) – Este campo se pondrá a 0 para indicar un splice-in (comienzo) y a 1 para indicar un splice-out (fin).

Bitrate (velocidad binaria) – En el caso de un splice-out, es la velocidad binaria media de la sesión. Este campo se expresa en bits por segundo (bit/s), incluida la tara del paquete de transporte de este canal.

PlayedDuration (duración de la reproducción) – En el caso de un splice-out, es el número de tics de reloj a 90 kHz que ha durado realmente la reproducción.

7.6 Mensajes de actividad

Una vez completada la inicialización, el servidor puede enviar mensajes **Alive_Request (petición de actividad)** para asegurarse de que el empalmador está aún activo. Cada mensaje **Alive_Response (respuesta de actividad)** contiene una indicación del estado del empalmador que se envía al servidor (estado del dispositivo). Si no ha habido actividad en la conexión TCP/IP durante los 60 segundos anteriores, deberá enviarse un mensaje Alive_Request.

7.6.1 Mensaje Alive_Request (petición de actividad)

El campo data( ) del mensaje **Alive_Request** contiene la estructura Alive_Request_Data que se indica en el cuadro 7-8.

Cuadro 7-8/J.280 – Alive_Request_Data

Sintaxis	Bytes	Tipo
Alive_Request_Data {		
time( )		
}		

time( ) – Es el reloj actual en tiempo universal (UTC) del dispositivo emisor, su valor en un momento lo más cercano posible al envío del mensaje. Lo utilizan el empalmador y el servidor para comprobar hasta qué punto están sincronizados los dos sistemas. No se pretende que de esta manera se puedan sincronizar correctamente los sistemas para hacer un empalme fiable, pero los implementadores pueden utilizarlo de la manera que prefieran. Véase en 8.4 la sintaxis de la estructura time( ).

7.6.2 Mensaje Alive_Response (respuesta de actividad)

El campo data( ) del mensaje **Alive_Response** contiene la estructura Alive_Response_Data que se indica en el cuadro 7-9.

Cuadro 7-9/J.280 – Alive_Response_Data

Sintaxis	Bytes	Tipo
Alive_Response_Data {		
State	4	uimsbf
SessionID	4	uimsbf
time( )		
}		

State (estado) – Describe el estado del canal de salida.

Cuadro 7-10/J.280 – Estados del mensaje Alive_Response

Estado	Descripción
0x00	No hay salida
0x01	En el canal primario
0x02	En el canal de inserción

SessionID – Es el **SessionID** de la inserción que se reproduce en ese momento. Sólo es válido cuando se tiene el valor de estado **(State)** = 0x02.

time( ) – Es el reloj actual en tiempo universal (UTC) del dispositivo emisor, su valor en un momento lo más cercano posible al envío del mensaje. Lo utilizan el empalmador y el servidor para comprobar hasta qué punto están sincronizados los dos sistemas. No se pretende que de esta manera se puedan sincronizar correctamente los sistemas para hacer un empalme fiable, pero los implementadores pueden utilizarlo como prefieran. Véase en 8.4 la sintaxis de la estructura time( ).

7.7 Mensajes de datos ampliados

Se trata de una estructura definida por el empalmador para enviar datos detallados sobre la reproducción al servidor. Una vez recibido el mensaje **SpliceComplete_Response** pueden extraerse los datos ampliados utilizando **ExtendedData_Request (petición de datos ampliados)**. Este mensaje tiene el mismo **SessionID** que la configuración de la sesión y el mensaje **SpliceComplete_Response**.

7.7.1 Mensaje ExtendedData_Request (petición de datos ampliados)

El campo data( ) de este mensaje contiene la estructura ExtendedData_Request_Data que se indica en el cuadro 7-11.

Cuadro 7-11/J.280 – ExtendedData_Request_Data

Sintaxis	Bytes	Tipo
ExtendedData_Request_Data {		
SessionID	4	uimsbf
ExtendedDataType	4	uimsbf
}		

SessionID – Es el **SessionID** de la sesión terminada.

ExtendedDataType (tipo de datos ampliados) – Es el tipo de datos que el empalmador solicita y se van a comunicar en el mensaje **ExtendedData_Response (respuesta de datos ampliados)**. Este valor puede ponerse a 0xFFFFFFFF para indicar el tipo de datos por defecto. Esta Recomendación reserva los valores entre 0x00000000 y 0x7FFFFFFF para normalización posterior. Los valores entre 0x80000000 y 0xFFFFFFFE se destina a la utilización exclusiva de los proveedores.

7.7.2 Mensaje ExtendedData_Response (respuesta de datos ampliados)

El servidor utilizará el campo **MessageSize (tamaño del mensaje)** para determinar la cantidad de datos que han de leerse en el mensaje **ExtendedData_Response**.

El campo data( ) de este mensaje contiene la estructura ExtendedData_Response_Data que se indica en el cuadro 7-12.

Cuadro 7-12/J.280 – ExtendedData_Response_Data

Sintaxis	Bytes	Tipo

ExtendedData_Response_Data {
SessionID
for (i=0;i<n;i++)
splice_API_descriptor( )
}
	4	uimsbf

SessionID – Es el **SessionID** para el cual estos datos son válidos.

splice_API_descriptor( ) – Descriptor del formato expuesto en 8.5, definido por el empalmador.

7.8 Mensajes de cancelación

El servidor puede enviar un mensaje **Abort_Request (petición de cancelación)** en cualquier momento, lo que hará que el empalmador vuelva inmediatamente al canal de inserción o al canal primario anulado. El empalmador enviará un mensaje **Abort_Response (respuesta de cancelación)** para acusar recibo de la **Abort_Request**. Se envía un **SpliceComplete_Response** con código de resultado 116 (inserción cancelada) si la **Abort_Request** ha causado un Splice-out de la inserción. Si no resultara necesario hacer un Splice-out, no se enviará ningún mensaje **SpliceComplete_Response**.

También se anularán todas las inserciones adosadas pendientes vinculadas por el campo **PriorSession** del mensaje **Splice_Request** al **SessionID** de un mensaje **Abort_Request**. Se enviará otro mensaje de error para cada **SessionID** cancelado. Considérese el siguiente ejemplo: aviso de tres inserciones en secuencia dentro de un bloque de tiempo: el primer evento está determinado por un valor tiempo; el segundo evento está vinculado al primer **SessionID** mediante **PriorSession**; el tercer evento está vinculado al segundo **SessionID** utilizando el mismo **PriorSession**. En este ejemplo, si se cancela el primer evento de inserción, también se cancelarán los dos eventos de inserción posteriores. El mensaje de cancelación no cancela ninguna inserción que utilice una conexión API diferente entre un servidor y el empalmador. Para el siguiente empalme en el canal primario es preciso que **PriorSession** en el mensaje de empalme sea 0xFFFFFFFF.

7.9 Mensaje Abort_Request (petición de cancelación)

El campo data( ) de este mensaje contiene la estructura Abort_Request_Data que se indica en el cuadro 7-13.

Cuadro 7-13/J.280 – Abort_Request Data

Sintaxis	Bytes	Tipo

Abort_Request_Data {
SessionID
}
	4	uimsbf

SessionID – Identificador de la sesión a cancelar, así como todas las sesiones posteriores vinculadas mediante el campo **PriorSession**.

7.10 Mensaje Abort_Response (respuesta de cancelación)

El mensaje **Abort_Response** no contiene datos e indica que se ha recibido el mensaje **Abort_Request**. Ese mensaje puede incluir un código de resultado, de ser necesario.

7.11 Peticiones de parámetros de configuración

El sistema puede comunicar la configuración actual de la conexión API, que incluye parte de la información de **Init_Request**. El mensaje **GetConfig_Request (petición de configuración)** no contiene datos adicionales.

7.11.1 Mensaje GetConfig_Request (petición de configuración)

El mensaje **GetConfig_Request** no contiene datos.

7.11.2 Mensaje GetConfig_Response (respuesta de configuración)

El campo data( ) de este mensaje contiene la estructura GetConfig_Response_Data que se indica en el cuadro 7-14.

Cuadro 7-14/J.280 – GetConfig_Response Data

Sintaxis	Bytes	Tipo

GetConfig_Response_Data {
ChannelName
Hardware_Config( )
TS_program_map_section( )
}
	32	String

ChannelName (nombre del canal) – Es el nombre lógico que se da al canal de salida de esta conexión.

Hardware_Config( ) (configuración de soporte físico) – Véase en 8.2 la sintaxis de la estructura Hardware_Config( ).

TS_program_map_section( ) (sección de correspondencia de programas del tren de transporte) – Se trata de toda la sección PMT del canal de salida, como se define en la Rec. UIT-T H.222.0 | ISO/CEI 13818-1. Si el empalmador modifica la tabla PMT, debe indicar este cambio al servidor con un código de resultado 128 en el mensaje **General_Response**.

7.12 Mensaje General_Response (respuesta general)

El mensaje General_Response se utiliza para transportar información asíncrona entre el servidor y el empalmador. No hay campo data( ) asociado con este mensaje. En él puede enviarse cualquier código de resultado. Este tipo de mensaje se utilizará normalmente para indicar cambios en la PMT del canal de salida o mensajes de petición no válidos.

8 Estructuras adicionales

8.1 Version (versión)

La estructura Version se utiliza para actualizar el número de versión en la API. En previsión de una evolución de la interfaz, la versión se especifica en los mensajes **Init_Request** e **Init_Response** para asegurarse de que el empalmador soporta la misma versión que el servidor.

Cuadro 8-1/J.280 – Version( )

Sintaxis	Bytes	Tipo
Version {		
Revision_Num	2	uimsbf
}		

Revision_Num (número de revisión) – Este campo es uno en esta versión.

El servidor y el empalmador deberían configurar y verificar este campo para garantizar que ambos componentes sean capaces de funcionar en la revisión que corresponda.

8.2 Hardware_Config

Esta estructura describe la interfaz de soporte físico entre el servidor y el empalmador. Es importante que el empalmador conozca exactamente dónde está conectado el servidor, para poder conocer la referencia del múltiplex. Un ejemplo de este enlace sería una conexión DVB-ASI entre el servidor y el empalmador.

Cuadro 8-2/J.280 – Hardware_Config( )

Sintaxis	Bytes	Tipo
Hardware_Config {		
Length	2	uimsbf
Chassis	2	uimsbf
Card	2	uimsbf
Port	2	uimsbf
Logical_Multiplex_Type	2	uimsbf
Logical_Multiplex		uimsbf
}		

Length (longitud) – Este campo da la longitud, en bytes, de la estructura según este campo.

Chassis (chasis) – Es un entero que indica a qué chasis del empalmador está conectado el múltiplex de inserción del servidor. Cuando la tarjeta esté etiquetada alfabéticamente, se indicará el número entero correspondiente (es decir, A – 1; B – 2; etc.).

Card (tarjeta) – Es un entero que indica la tarjeta del empalmador a la que está conectado el múltiplex de inserción del servidor. Cuando la tarjeta esté etiquetada alfabéticamente, se indicará el número entero correspondiente (es decir, A – 1; B – 2; etc.).

Port (puerto) – Es el número de puerto del soporte físico al que está conectado el múltiplex de inserción del servidor.

Logical_Multiplex_Type (tipo de múltiplex lógico) – Valor que se extrae del cuadro 8-3.

Cuadro 8-3/J.280 – Tipo de múltiplex lógico

Tipo	Longitud	Nombre	Descripción
0x0000	0	No utilizado	No se necesita el campo **Logical_Multiplex (múltiplex lógico)** para identificar el múltiplex.
0x0001	Variable	Definido por el usuario	La utilización del campo **Logical_Multiplex** no viene definida por esta Recomendación, y debe acordarse entre el empalmador y el servidor.
0x0002	6	Dirección MAC	El campo **Logical_Multiplex** contiene la dirección de control de acceso a medios IEEE del múltiplex, como una dirección de 6 bytes.
0x0003	6	Dirección IPv4	Los 4 bytes más significativos del campo **Logical_Multiplex** contienen la dirección del múltiplex en Protocolo Internet (IP), y los 2 bytes restantes contienen el número de puerto IP en que puede encontrarse el múltiplex.
0x0004	18	Dirección IPv6	Los 16 bytes más significativos del campo **Logical_Multiplex** contienen la dirección del múltiplex en Protocolo Internet (IPv6), y los 2 bytes contienen el número de puerto IP donde puede encontrarse el múltiplex.
0x0005	5	Dirección ATM	El campo **Logical_Multiplex** contiene las coordenadas del circuito de modo de transferencia asíncrono (ATM) por el que se transporta el múltiplex. Los 2 bytes más significativos del campo de **múltiplex lógico** contienen el identificador de trayecto virtual (VPI, *virtual path identifier*). Los siguientes dos bytes contienen el identificador de canal virtual (VCI, *virtual channel identifier*) del circuito. El byte menos significativo contiene el número de capa AAL.
0x0006	Variable	Dirección IPv4 con soporte SPTS	Véase descripción después de este cuadro.
0x0007	Variable	Dirección IPv6 con soporte SPTS	Véase descripción después de este cuadro.
0x0008-0xFFFF	Variable	Reservado	Reservado para la normalización futura.

Logical_Multiplex – Si **Port (puerto)** transporta varios múltiplex de inserción hacia una sola entrada, este campo permite al empalmador determinar cuál utilizar al realizar el empalme procedente del servidor. El significado y el formato de este campo están definidos por el campo **Logical_Multiplex_Type (tipo de múltiplex lógico)**. En caso de que se requiera una definición no normalizada de **Logical_Multiplex**, se pondrá a 1 el campo **Logical_Multiplex_Type** para indicar definido por el usuario.

Type 0x0006 – IPv4 address (tipo 0x0006 – dirección IPv4) con soporte para flujo único de transporte de programación (SPTS)

Los servidores VoD y Ad utilizan el tipo 0x0006, cuando volver a hacer corresponder PID resulta poco práctico o no deseable. En estos casos convendría utilizar un SPTS por puerto UDP.

Cuadro 8-4/J.280 – Estructura del tipo 0x0006

Sintaxis	Bytes	Tipo
Type 0x0006 structure {		
number_of_destination_ips	1	uimsbf
for (j=0; j< **number_of_destination_ips**; j++) {		
dest_ip_address	4	uimsbf
}		
number_of_source_ips	1	uimsbf
for (j=0; j< **number_of_source_ips**; j++) {		
source_ip_address	4	uimsbf
}		
base_port	2	uimsbf
number_of_ports	1	uimsbf
}		

number_of_destination_ips (número de ip de destino) – Especifica cuántas **dest_ip_address** siguen. La gama válida va de 1 a 32.

dest_ip_address (dirección ip de destino) – La dirección IPv4 que el empalmador debe utilizar para el contenido asociado con el empalme.

number_of_source_ips (número de ip de origen) – Especifica cuántas **source_ip_address** siguen. La gama válida va de 0 a 32.

source_ip_address (direcciones ip de origen) – La dirección o direcciones de origen IPv4 que el empalmador puede utilizar en un empalme IGMP V3 para la **dirección o direcciones ip de destino** multidistribuidas asociadas.

base_port (puerto de base) – El puerto UDP inicial que el empalmador debe utilizar para el contenido asociado con una Splice_Request con time( ) especificado. IANA debe asignar la gama de puertos de base de UDP.

number_of_ports (número de puertos) – Este byte contiene el número de puertos contiguos para su reserva. El valor del number_of_ports puede ir de 1 a 4 e incluye el puerto de base. Los números permitidos de puertos quedan determinados sucesivamente, por el puerto de base+1, seguido por el puerto de base+2, seguido por el puerto de base+3.

Todas las Splice_Requests que utilizan time( ) deben emplear base IPv4 Address:Port a menos de que se utilice el port_selection_descriptor( ). La primera Splice_Request de un espacio disponible debe utilizar time( ). Las sesiones siguientes del mismo espacio disponible que también utilizan time( ) deben utilizar igualmente base IPv4 Address:Port, a menos de que se utilice el port_selection_descriptor( ). La siguiente y ulteriores splice_requests que utilizan PriorSession en lugar de time( ), deben utilizar el IP de base y el puerto+1, y a continuación el puerto+2, etc., hasta que se utilice el número solicitado de puertos que deben volver al puerto de base para efectuar la siguiente splice_request.

El port_selection_descriptor( ) puede ser utilizado en cualquier comando Splice_Request que tenga un equipo configurado con Logical_Multiplex type 0x0006 para modificar las operaciones por defecto de los puertos.

El puerto puede ser cualquier combinación de IPv4 Address:Port unidistribuida o multidistribuida. El empalmador debe desempeñar un empalme IGMP o un IP multidistribuido.

Type 0x0007 – Dirección IPv6 (tipo 0x0007 – dirección IPv6) con soporte para flujo único de transporte de programación (SPTS)

Los servidores VoD y Ad utilizan Type 0x0007, cuando volver a hacer corresponder PID resulta poco práctico o no es deseable. En estos casos convendría utilizar un SPTS por puerto UDP.

Cuadro 8-5/J.280 – Estructura del tipo 0x0007

Sintaxis	Bytes	Tipo
Type 0x0007 structure {		
number_of_destination_ips	1	uimsbf
for (j=0; j< **number_of_destination_ips**; j++) {		
dest_ip_address	16	uimsbf
}		
number_of_source_ips	1	uimsbf
for (j=0; j< **number_of_source_ips**; j++) {		
source_ip_address	16	uimsbf
}		
base_port	2	uimsbf
number_of_ports	1	uimsbf
}		

number_of_destination_ips (número de ip de destino) – Especifica cuántas **dest_ip_address** siguen. La gama válida va de 1 a 32.

dest_ip_address (dirección ip de destino) – La dirección IPv6 que el empalmador debe utilizar para el contenido asociado con el empalme.

number_of_source_ips (número de ip de origen) – Especifica cuántas **source_ip_address** siguen. La gama válida va de 0 a 32.

source_ip_address (direcciones IP de origen) – La dirección o direcciones de origen IPv6 que el empalmador puede utilizar en un empalme MLD V2 para la **dirección o direcciones ip de destino** multidistribuidas asociadas.

base_port (puerto de base) – El puerto UDP que el empalmador debe utilizar para el contenido asociado con una Splice_Request con time( ) especificado. IANA debe asignar la gama de puertos de base de UDP.

number_of_ports (número de puertos) – Este byte contiene el número de puertos contiguos para su reserva. El valor del number_of_ports puede ir de 1 a 4 e incluye el puerto de base. Los números permitidos de puertos quedan determinados sucesivamente, por el puerto de base, seguido por el puerto de base+1, seguido por el puerto de base+2, seguido por el puerto de base+3.

Todas las Splice_Requests que utilizan time( ) deben utilizar IPv6 Address:Port de base a menos de que se utilice el port_selection_descriptor( ). La primera Splice_Request de un espacio disponible debe utilizar time( ). Las sesiones siguientes del mismo espacio disponible que también utilizan time( ) deben utilizar igualmente IPv6 Address:Port de base, a menos de que se utilice el port_selection_descriptor( ). La siguiente y ulteriores splice_requests que utilizan PriorSession en lugar de time( ), deben utilizar el IP de base y el puerto+1, y a continuación el puerto+2, etc., hasta que se utilice el número solicitado de puertos que deben volver al puerto de base para efectuar la siguiente splice_request.

El port_selection_descriptor( ) puede ser utilizado en cualquier comando Splice_Request que tenga un equipo configurado con Logical_Multiplex type 0x0007 para modificar las operaciones por defecto de los puertos.

El puerto puede ser cualquier combinación de IPv6 Address:Port unidistribuida o multidistribuida. El empalmador debe desempeñar un empalme MLD o un IP distribuido.

8.3 splice_elementary_stream( ) (tren elemental del empalme)

Los identificadores de paquete (PID) individualizan partes del tren de transporte, vídeo, audio, datos, etc. Esta estructura se utiliza para describir uno de los elementos del programa en el MPTS. El mensaje **Splice_Request** puede contener una estructura splice_elementary_stream( ) para cada uno de los componentes del tren de transporte (excepto para el PID de la PCR). Los **StreamType**s (**tipos de tren**) se basan en las definiciones de la tabla PMT MPEG.

Esta Recomendación no define la correspondencia entre varios PID de audio/vídeo/datos y un PID de salida. Tampoco define el comportamiento del empalmador cuando haya una diferencia entre el canal de inserción y el canal primario en el número de pistas de audio (hay otras o faltan algunas).

Cuadro 8-6/J.280 – splice_elementary_stream( )

Sintaxis	Bytes	Tipo
splice_elementary_stream {		
Length	1	uimsbf
PID	2	uimsbf
StreamType	2	uimsbf
AvgBitrate	4	uimsbf
MaxBitrate	4	uimsbf
MinBitrate	4	uimsbf
HResolution	2	uimsbf
VResolution	2	uimsbf
for (i=0;i<N;i++)		
descriptor( )		
}		

Se requiere el PID de la PCR.

Length (longitud) – Longitud total de la estructura splice_elementary_stream( ).

PID (identificador de paquete) – Es el número PID que se está utilizando. Se trata de un campo de 2 bytes (16 bits) que contendrá el PID de 13 bits alineados a la derecha como un entero de 16 bits (0x0000 a 0x1FFF).

StreamType (tipo de tren) – Es el tipo de PID (audio, vídeo, etc.). Este número corresponde a la especificación de la PMT de la Rec. UIT-T H.222.0 | ISO/CEI 13818-1.

AvgBitrate (velocidad binaria media) – Es la velocidad binaria media para todo el contenido de este PID (en bits por segundo, bit/s), a la que se ha codificado el contenido. Este valor se pone a 0xFFFFFFFF si se desconoce la velocidad binaria de codificación.

MaxBitrate (velocidad binaria máxima) – Es la velocidad binaria máxima de este PID. Se pone a 0xFFFFFFFF si se desconoce la velocidad binaria.

MinBitrate (velocidad binaria mínima) – Es la velocidad binaria mínima para este PID. Se pone a 0xFFFFFFFF si se desconoce la velocidad binaria.

HResolution (resolución horizontal) – Es la anchura en número de píxels de las imágenes de vídeo que utilizan este PID. Si el PID no contiene imágenes de vídeo, o si el servidor no puede proporcionar este valor, se pondrá a 0xFFFF.

VResolution (resolución vertical) – Es la altura en número de píxels de las imágenes de vídeo que utilizan este PID. Si el PID no contiene imágenes de vídeo, o el servidor no puede proporcionar este valor, se pondrá a 0xFFFF.

descriptor( ) – Puede ser cualquier descriptor válido que se utilice en una PMT. Para múltiples PID de audio, es necesario utilizar los descriptores de lenguaje definidos en la Rec. UIT–T H.222.0 |ISO/CEI 13818-1.

8.4 Definición del campo time( )

La estructura time se utiliza para definir diversos instantes de empalme en esta Recomendación.

Cuadro 8-7/J.280 – time( )

Sintaxis	Bytes	Tipo
time {		
Seconds	4	uimsbf
MicroSeconds	4	uimsbf
}		

Seconds (segundos) – Segundos pasados desde las 12:00 AM del 1 de enero de 1970, tiempo universal (UTC).

MicroSeconds (microsegundos) – Diferencia del campo **Seconds** en microsegundos.

8.5 Definición del campo splice_API_descriptor( ) (descriptor API de empalme)

Se trata de un modelo para añadir descriptores en cualquier mensaje definido en esta Recomendación. Pueden utilizar descriptores los mensajes **Splice_Request**, **ExtendedData_Response** e **Init_Request**. La utilización de descriptores en los mensajes definidos por la presente Recomendación es facultativa. En el cuadro 8-8 se expone el formato general de los descriptores utilizados en esta norma.

Cuadro 8-8/J.280 – splice_api_descriptor( )

Sintaxis	Bytes	Tipo
splice_API_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_API_Identifier	4	uimsbf
for (i=0;i<n;i++)		
Private_Byte	1	uimsbf
}		

Splice_Descriptor_Tag (etiqueta de descriptor de empalme) – Es un valor entre 0x00 y 0xFF que indica el descriptor que se está utilizando. Los valores de etiqueta 0x00 a 0xFF se reservan para esta Recomendación. Los proveedores pueden utilizar un **Splice_API_Identifier (identificador API de empalme)** propio para disponer de una gama de etiquetas más amplia y de un método más robusto para añadir descriptores propios.

Descriptor_Length (longitud del descriptor) – Indica la longitud, en bytes, del descriptor que sigue a este campo. Los descriptores están limitados a 256 bytes, por lo que el valor de este campo se limita a 254.

Splice_API_Identifier (identificador API de empalme) – Es el identificador de la organización que ha definido este descriptor. El identificador de todos los descriptores de la presente Recomendación es 0x53415049 (ASCII "SAPI"). Se ha elegido este valor para evitar coincidencias con descriptores de cualquier otro identificador conocido.

Private_Byte (byte privado) – El resto del descriptor está dedicado a campos de datos requeridos por el descriptor que se define.

8.5.1 Definiciones del campo playback_descriptor( ) (descriptor de reproducción)

playback_descriptor( ) es una implementación del splice_API_descriptor( ) que se utiliza en el mensaje **Splice_Request**.

La velocidad de reproducción, definida como la velocidad media del canal de salida durante un periodo de un segundo es uno de los criterios de cancelación. Se recomienda que la ventana de promediación de la velocidad se desplace por intervalos de un segundo o menos.

Cuadro 8-9/J.280 – playback_descriptor( )

Sintaxis	Bytes	Tipo
playback_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_API_Identifier	4	uimsbf
BitrateRule	1	uimsbf
MinPlaybackRate	4	uimsbf
}		

Splice_Descriptor_Tag – 0x01.

DescriptorLength – 0x09.

Splice_API_Identifier – 0x53415049, ASCII "SAPI".

BitrateRule (regla de velocidad binaria) – Bandera utilizada para indicar las reglas aplicables a **MinPlaybackRate (velocidad de reproducción mínima)**.

Cuadro 8-10/J.280 – Valores de BitrateRule

BitrateRule	Descripción
0x00	No tener en cuenta el valor de **MinPlaybackRate**.
0x01	Devolver inmediatamente un código resultado 127 utilizando el mensaje **General_Response**, si la velocidad de reproducción es inferior a **MinPlaybackRate**, pero no cancelar.
0x02	Cancelar si la velocidad de reproducción es inferior a **MinPlaybackRate**.
0x03	Cancelar la sesión antes de Splice-in si el empalmador determina que no conseguirá alcanzar el valor definido por **MinPlaybackRate**. El empalmador enviará **SpliceComplete_Response** o **General_Response** con un código de resultado 127.

MinPlaybackRate (velocidad de reproducción mínima) – Es la velocidad binaria media mínima del canal de salida calculada para todos los intervalos de un segundo de la duración del empalme, el límite inferior de reproducción por debajo del cual se aplicará **BitrateRule**. Poner este valor a 0 indica que no hay una velocidad mínima.

8.5.2 Definiciones del campo muxpriority_descriptor( ) (descriptor de prioridad del múltiplex)

muxpriority_descriptor( ) es una implementación de splice_API_descriptor( ) que se utiliza en el mensaje **Splice_Request**.

Cuadro 8-11/J.280 – muxpriority_descriptor( )

Sintaxis	Bytes	Tipo
muxpriority_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_API_Identifier	4	uimsbf
MuxPriorityValue	1	uimsbf
}		

Splice_Descriptor_Tag – 0x02

DescriptorLength – 0x05

Splice_API_Identifier – 0x53415049, ASCII "SAPI".

MuxPriorityValue (valor de prioridad del múltiplex) – Un número entre 1 y 10 (siendo 1 el valor más bajo, 5 el valor medio y 10 el más alto). Este número modifica el **MuxPriorityValue** almacenado para canal primario en el empalmador. Un **MuxPriorityValue** de 5 no modificará la prioridad de los canales de salida. Un **MuxPriorityValue** inferior a 5 reducirá el nivel de prioridad del canal de salida y un **MuxPriorityValue** superior a 5 aumentará la prioridad de los canales de salida.

Al utilizar **MuxPriorityValue** no se garantiza que el contenido se reproducirá con un nivel específico de calidad. El efecto real de **MuxPriorityValue** depende de toda la configuración del múltiplex empalmado y de la reducción de velocidad binaria total del múltiplex que el empalmador tenga que hacer en un momento dado. También depende del funcionamiento del empalmador y, por tanto, se trata de un campo dependiente del proveedor del empalmador.

8.5.3 Definiciones del campo missing_Primary_Channel_action_descriptor( ) (falta descriptor de acción del canal primario)

missing_Primary_Channel_action_descriptor( ) es una implementación de splice_API_descriptor( ) que se utiliza en el mensaje **Init_Request**.

Si el canal primario se ha cortado por cualquier motivo durante una inserción, es posible que el decodificador transmita una trama congelada de la última trama insertada al final de la inserción. Este descriptor permite indicar al empalmador que inserte un vídeo en negro y un audio mudo para vaciar la memoria intermedia del decodificador, en caso de que el canal primario ya no esté presente cuando normalmente habría de ser la fuente de audio y vídeo.

Cuadro 8-12/J.280 – missing_Primary_Channel_action_descriptor ( )

Sintaxis	Bytes	Tipo
missing_Primary_Channel_action_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_API_Identifier	4	uimsbf
MissingPrimaryChannelAction	1	uimsbf
}		

Splice_Descriptor_Tag – 0x03

DescriptorLength – 0x05

Splice_API_Identifier – 0x53415049, ASCII "SAPI".

MissingPrimaryChannelAction (falta acción del canal primario) – Este parámetro tiene tres valores posibles: 0, 1, y 2. El valor 0 significa que no se ha de hacer nada. El valor 1 significa que ha de insertarse una trama I en negro y una trama de audio mudo. El valor 2 significa que ha de continuar transmitiéndose el vídeo en negro y el audio mudo hasta el retorno de la señal primaria.

8.5.4 Definiciones del campo port_selection_descriptor( )

El port_selection_descriptor( ) es una implementación del API_descriptor( ) que debe utilizarse únicamente en el mensaje **Splice_Request** cuando Logical_Multiplex type 0x0006 ó 0x0007 se utiliza en la configuración del equipo. Si durante una secuencia de inserciones el servidor envía un port_selection_descriptor( ), el servidor debe seguir enviando el port_selection_decriptor( ) hasta que tenga lugar la siguiente Splice_Request basada en el tiempo.

El port_selection_descriptor( ) puede utilizarse para modificar la operación por defecto de los puertos o seleccionar una nueva combinación IPv4 o IPv6 Address:Port configurada dinámicamente.

El empalmador debe configurar dinámicamente un puerto de destino si la **ps_ip_address** no se ha definido en la configuración del equipo. Si la **ps_ip_address** está multidistribuida, el empalmador debe enviar una petición de empalme de IGMP o de empalme MLD dentro de los 400 milisegundos transcurridos después de la llegada del mensaje Splice_Request. El periodo de latencia para configurar un grupo multidistribuido debe ser inferior a 2 segundos y se obtiene del siguiente modo:

– Tres segundos para la llegada del mensaje Splice_Request (véase 7.5).

– Menos el tiempo de inicio del flujo igual a 600 milisegundos (véase 7.5).

– Menos los 400 milisegundos necesarios para que el empalmador expida la petición de empalme IGMP o de empalme MLD.

Cuadro 8-13/J.280 – IPv4 port_selection_descriptor ( )

Sintaxis	Bytes	Tipo
port_selection_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_API_Identifier	4	uimsbf
ps_ip_address	4	uimsbf
ps_port	2	uimsbf
ps_number_of_source_ip	1	uimsbf
for (j=0; j< **ps_number_of_source_ip**; j++) {		
ps_source_ip_address	4	uimsbf
}		
}		

Splice_Descriptor_Tag (etiqueta de descriptor de empalme) – 0x04

Descriptor_Length (longitud del descriptor) – Variable

Splice_API_Identifier (identificador de empalme API) – 0x53415049, ASCII "SAPI".

ps_ip_address – La dirección de protocolo Internet IPv4 que el empalmador debe utilizar respecto al contenido asociado con el empalme. Si esta combinación de address:port es diferente de la dirección consignada en el cuadro relativo a Logical_Mux_Type 0x0006, debe considerarse como una petición de configuración dinámica de puerto.

port_ps – El puerto UDP que el empalmador debe utilizar en relación con el contenido asociado con el empalme. Este número de puerto debe prevalecer sobre el método de selección automática de puerto de Logical_Multiplex_Type 0x0006.

ps_number_of_source_ip (número de ps de ip de origen) – Especifica cuántas **ps_source_ip_address** siguen. La gama válida va de 0 a 32.

ps_source_ip_address (dirección ip de origen de ps) – La fuente de dirección o direcciones IPv4 que el empalmador debe utilizar en un empalme IGMP V3 para la **ps_ip_address** multidistribuida asociada.

Cuadro 8-14/J.280 – IPv6 port_selection_descriptor ( )

Sintaxis	Bytes	Tipo
port_selection_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_API_Identifier	4	uimsbf
ps_ip_address	16	uimsbf
ps_port	2	uimsbf
ps_number_of_source_ip	1	uimsbf
for (j=0; j< **ps_number_of_source_ip**; j++) {		
ps_source_ip_address	16	uimsbf
}		
}		

Splice_Descriptor_Tag – 0x05

Descriptor_Length – Variable

Splice_API_Identifier – 0x53415049, ASCII "SAPI".

ps_ip_address – La dirección de protocolo Internet IPv6 que el empalmador debe utilizar respecto al contenido asociado con el empalme. Esta combinación de address:port es diferente que la dirección consignada en el cuadro relativa a Logical_Mux_Type 0x0007, debe considerarse como una petición de configuración dinámica de puerto.

port_ps – El puerto UDP que el empalmador debe utilizar en relación con el contenido asociado con el empalme. Este número de puerto debe prevalecer sobre el método de selección automática de puerto de Logical_Multiplex_Type 0x0007.

ps_number_of_source_ip (número ps de ip de origen) – Especifica cuántas **ps_source_ip_address** siguen. La gama válida va de 0 a 32.

ps_source_ip_address (dirección ip de origen de ps) – La fuente de dirección o direcciones IPv6 que el empalmador debe utilizar en un empalme IGMP V3 para la **ps_ip_address** multidistribuida asociada.

9 Sincronización temporal

Es necesario establecer una sincronización temporal debido a la diferencia de tiempo entre el servidor y el empalmador. El retardo en un mensaje TCP/IP es bastante impredecible y en parte depende de otros equipos de la red. Al sincronizar los dispositivos se tiene en cuenta este tiempo y se consigue un empalme muy exacto sin problemas para los retardos normales de la red. Se puede, pero no es obligatorio, utilizar el protocolo de tiempo de red (NTP, *network time protocol*) para mantener el servidor y el empalmador sincronizados. Es probable que los servidores ya realicen algún tipo de sincronización, constituyéndose así el servicio NTP y siendo el empalmador un cliente NTP. También puede utilizarse un servidor NTP de sistema anfitrión común de red, puesto que ya existirá normalmente en la cabecera del cable con infraestructura de red.

Un sistema de sincronización temporal debe mantener una diferencia de ±15 ms entre el empalmador y el servidor. Esta exactitud de la sincronización temporal (es decir, diferencia de una trama de vídeo) se considera suficiente para el funcionamiento adecuado del servidor-empalmador conforme a esta Recomendación. El sistema puede utilizar mensajes **Alive_Request/Alive_Response** para detectar si dos dispositivos están adecuadamente sincronizados y avisar al operador si se pierde la sincronización.

El tren de bits que representa el canal primario está sometido a diversos retardos: el empalme en sentido ascendente, los enlaces de satélite y otros procesos de transmisión y acondicionamiento, que pueden variar entre milisegundos y segundos. De los valores de la PCR transportados en los trenes de transporte MPEG-2 puede obtenerse una referencia temporal del tren para compensar estos retardos. No obstante, estos retardos no afectan a la exactitud del mensaje de aviso incorporado en el canal primario. El mensaje de aviso utiliza la PCR para indicar el tiempo exacto de inserción, por lo que mantiene la exactitud original del contenido.

El servidor que proporciona el contenido del canal de inserción sólo conoce el tiempo del reloj (UTC), y ha sido programado con ventanas de inserción definidas según el tiempo del reloj. Ahora bien, es el empalmador el que debe indicarle en qué momento exactamente ha de empezar a difundir el contenido.

Cuando el empalmador recibe un tren de bit de programa, dicho tren ya ha sufrido todos los retardos. El empalmador puede tomar la PCR y relacionarla con el tiempo de reloj, y enviar a continuación un mensaje al servidor en el que se especifique el instante UTC exacto en el que ha de comenzar a difundir el contenido. El canal de inserción del servidor llega entonces al empalmador

exactamente sincronizado con el canal primario, y puede conseguirse un empalme perfecto. Todos los retardos adicionales que ocurran dentro del empalmador son irrelevantes puesto que los trenes binarios de entrada estaban sincronizados.

10 Temporización del sistema

10.1 Flujo de señales para un empalme de inserción de programa digital (DPI)

En las figuras 4 y 5 se muestran detalles específicos de la utilización y ordenamiento de los diversos mensajes que permite esta API. La utilización efectiva de los mensajes API no se limita necesariamente a estos ejemplos.

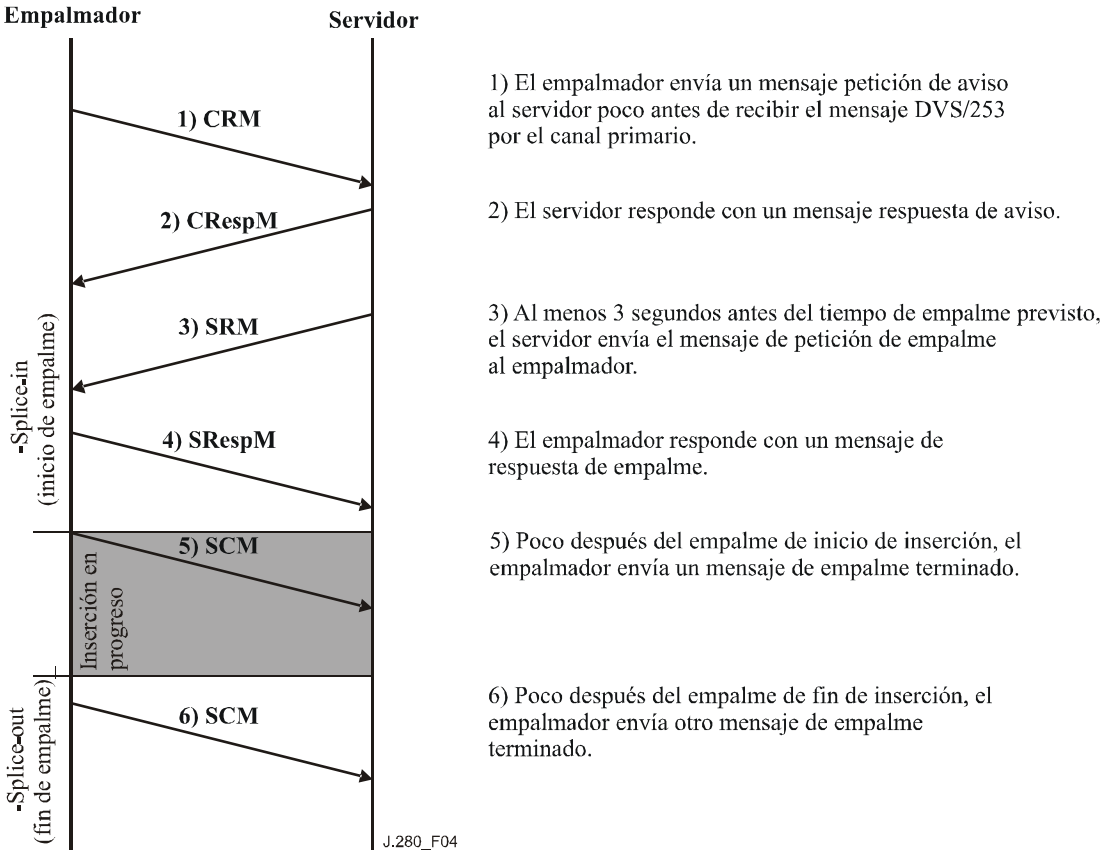


Figura 4/J.280 – Empalme en un sólo evento

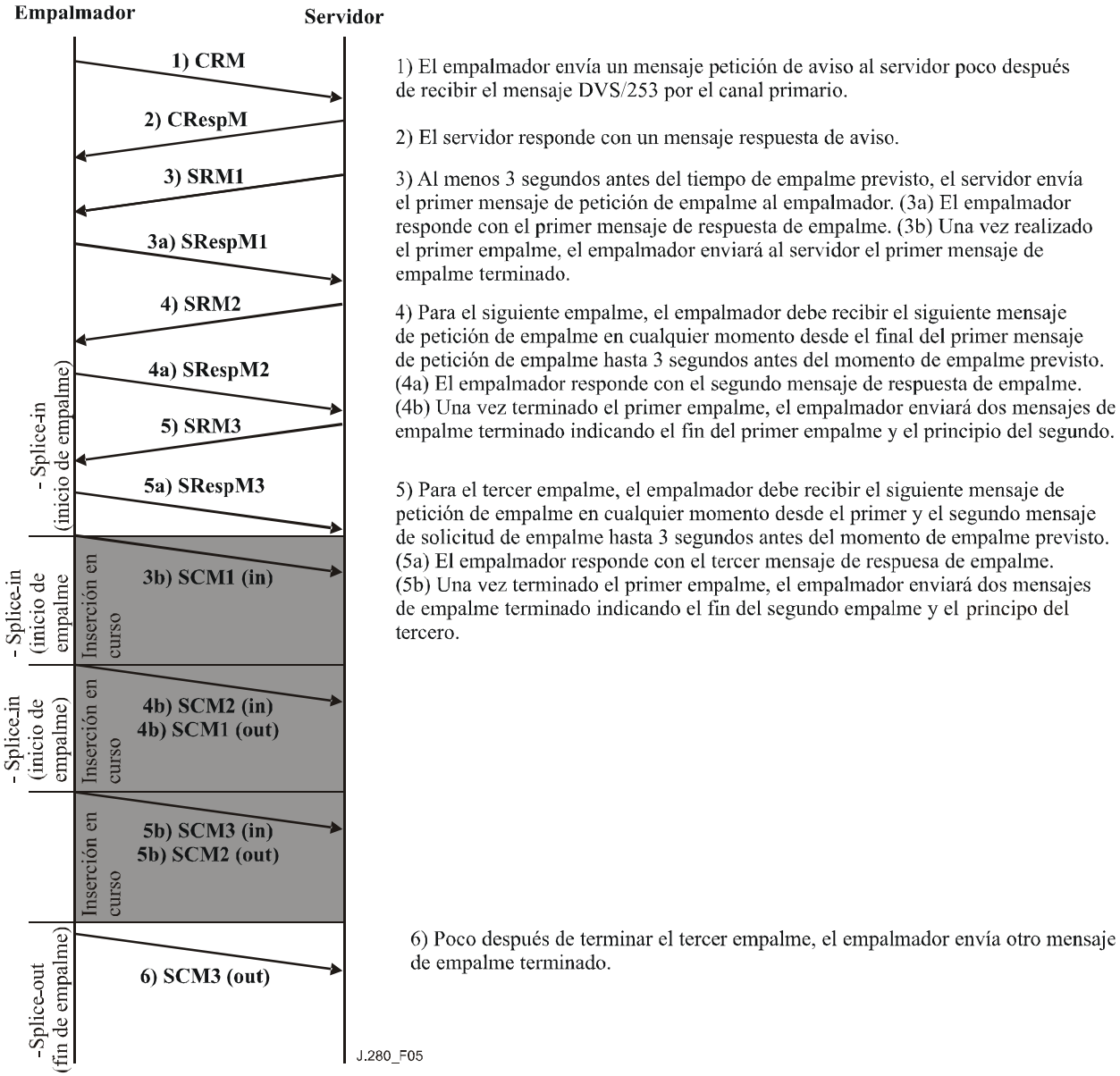


Figura 5/J.280 – Empalme en varios eventos

10.2 Instante de iniciación de empalme de inserción de programa digital (DPI)

En la figura 6 se da la temporización exacta de los eventos que preceden la inserción de un programa (o anuncio). Los tiempos en una situación real pueden variar de la temporización que se muestra en la figura. Este intervalo de tiempo vale para las condiciones de arbitraje de prioridad que se indican en 6.2. También se muestra el funcionamiento con mensajes de aviso J.181.

En la figura, las líneas negras gruesas indican el flujo de información MPEG por el canal primario y el canal de inserción. Las líneas negras en trazo fino indican que la información MPEG no fluye en ese momento o es irrelevante (es decir, no se ha seleccionado para salir por el canal de salida).

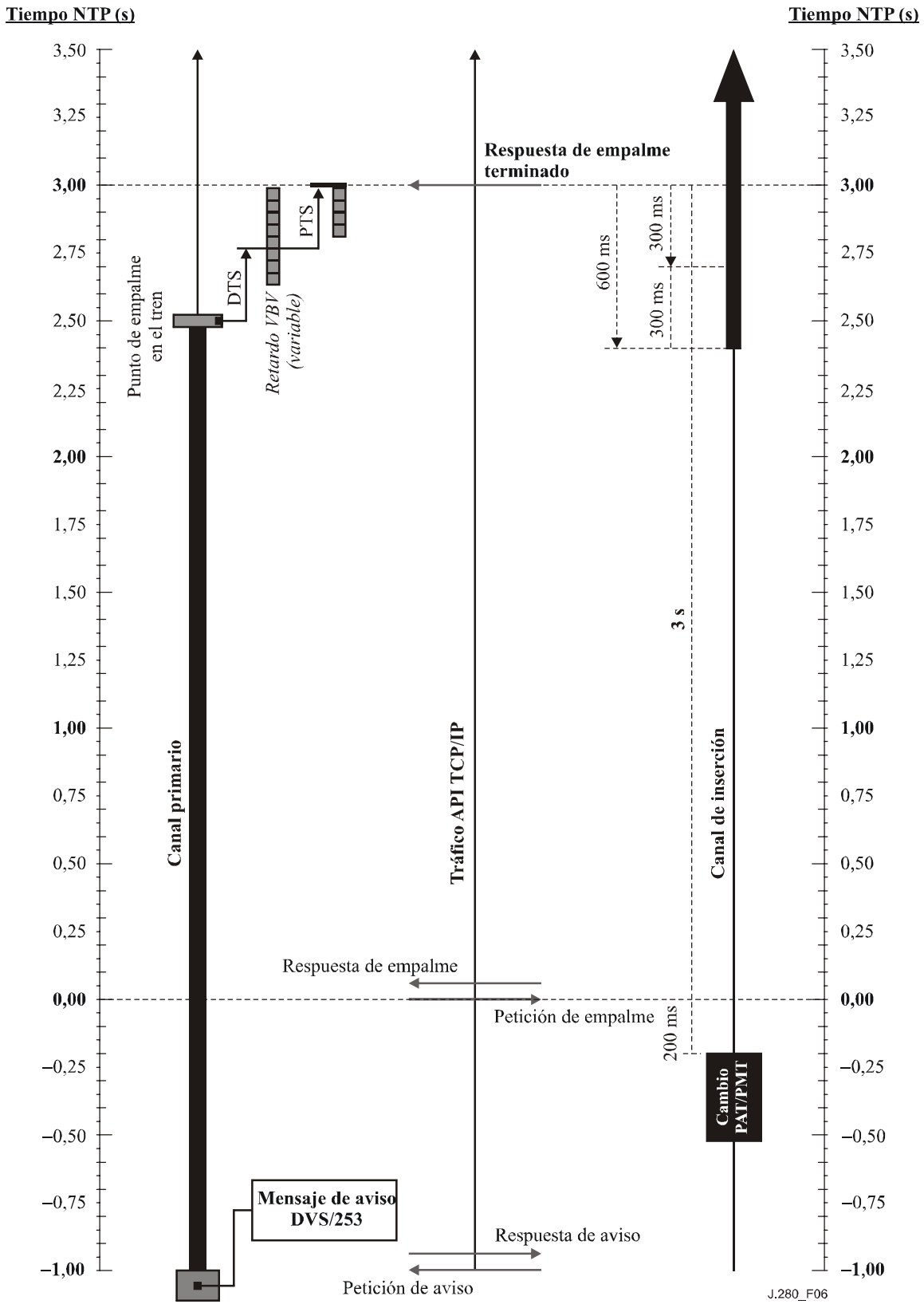


Figura 6/J.280 – Instante de iniciación de empalme DPI

Apéndice I

Códigos de resultado

Resultado	Nombre del resultado	Descripción	Mensaje de respuesta
100	Respuesta de éxito		Todos
101	Fallo desconocido		Todos
102	Versión no válida	El servidor y el empalmador están utilizando distintas versiones de esta API.	**Init_Response**
103	Acceso denegado	Puede haber un problema de licencia.	**Init_Response**
104	**ChannelName** no válido/desconocido	Posible error de configuración.	**Init_Response**
105	Conexión física no válida	Posible error de configuración.	**Init_Response**
106	No se encuentra la configuración	El empalmador es incapaz de determinar la configuración para esta conexión.	**GetConfig_Response**
107	Configuración no válida	Uno de los parámetros, o varios, de la configuración de esta conexión no son válidos.	**GetConfig_Response**
108	Empalme fallido – Fallo desconocido		**SpliceComplete_Response**
109	Colisión de empalme	Ya se ha configurado un empalme de prioridad igual o superior.	**Splice_Response** **SpliceComplete_Response**
110	Canal de inserción no encontrado	Se notificará este error si no hay un canal de inserción al principio del empalme.	**SpliceComplete_Response**
111	Canal primario no encontrado	Se notificará este error si no se encuentra el canal primario en los momentos Splice-in o Splice-out.	**SpliceComplete_Response**
112	**Splice_Request** demasiado tarde	El mensaje **Splice_Request** no se ha recibido con suficiente antelación (3 segundos) para que el empalmador inicie el empalme.	**Splice_Response**
113	Punto de empalme no encontrado	El empalmador no ha podido encontrar el punto válido para empalmar en el canal primario.	**SpliceComplete_Response**
114	Cola de empalme llena	Hay demasiados mensajes **Splice_Request** pendientes.	**Splice_Response**
115	Posibles problemas en la sesión de reproducción	El empalmador ha detectado discrepancias de vídeo o audio que pueden afectar a la reproducción.	**SpliceComplete_Response**
116	Inserción cancelada	Un mensaje **Abort_Request** ha causado un fin de empalme.	**SpliceComplete_Response**

Resultado	Nombre del resultado	Descripción	Mensaje de respuesta
117	Mensaje de aviso no válido	El empalmador o el servidor no han podido analizar sintácticamente el mensaje de aviso.	**General_Response** **Cue_Response**
118	Dispositivo de empalme inexistente	No se encuentra **SplicerName**.	**Init_Response**
119	**Init_Request** rechazada	El empalmador no permite al servidor la conexión.	**Init_Response**
120	**MessageID** desconocido	Se utiliza Splicing_API_Message para enviar la respuesta al emisor. Se devuelve el **MessageID** desconocido.	Todos
121	**SessionID** no válido	El empalmador no conoce la **SessionID** especificada.	**Splice_Response** **Abort_Response** **ExtendedData_Response**
122	Sesión no terminada	El empalmador no ha podido reproducir toda la duración. Esto incluye los casos en que el servidor no ha proporcionado contenido suficiente.	**SpliceComplete_Response**
123	Datos del mensaje de petición no válidos ( )	El empalmador o el servidor no han podido analizar sintácticamente un campo en el mensaje de petición. Se notifica la posición de este campo no válido en el campo **Result_Extension** de Splicing_API_Message.	Todos
124	Descriptor no está en la configuración	El empalmador no entiende ni tiene configurado actualmente el descriptor solicitado.	Respuesta a todos los mensajes que aceptan descriptores.
125	Canal anulado	Este código de resultado se utiliza para indicar que la inserción que se está reproduciendo ha sido anulada con un estado Splice-out o que se ha retomado con un estado Splice-in.	**SpliceComplete_Response**
126	Comienzo temprano del canal de inserción	Este error puede enviarse si el canal de inserción ha comenzado demasiado temprano y el empalmador no ha podido determinar el momento exacto de inicio del tren de inserción.	**SpliceComplete_Response**
127	Velocidad de reproducción por debajo del umbral	Véanse más detalles en la cláusula correspondiente playback_descriptor().	**SpliceComplete_Response**

Resultado	Nombre del resultado	Descripción	Mensaje de respuesta
128	Cambio de PMT	Se utiliza para indicar al servidor que se ha modificado la PMT de este canal primario.	**General_Response**
129	Tamaño del mensaje no válido	El mensaje no tenía la longitud correcta especificada en esta Recomendación.	Todos
130	Sintaxis del mensaje no válida	Los campos definidos por esta Recomendación no se encuentran dentro de la gama válida.	Todos
131	Error de colisión de puerto	El empalmador no ha podido utilizar la combinación IP:port especificada que se solicitó. La combinación está en uso o no es válida en este empalmador.	**Init_Response** **General_Response**
NOTA – Todos los códigos de resultado pueden utilizarse en el mensaje **General_Response**.			

Apéndice II

Ejemplo de utilización de Logical_Multiplex Type 0x0006 y del port_selection_descriptor( )

II.1 Ejemplo informativo 1

El siguiente ejemplo ilustra la utilización de múltiples Splice_Requests en secuencia y el aumento del número de puertos entre las siguientes peticiones cuando no están presentes port_selection_descriptors. (Véase 8.2.)

Todos los puertos se ponen estáticamente en Init_Request.

Base IP:Port = 192.168.134.9:2000

Número de puertos = 4

Los siguientes eventos tienen lugar de manera secuencial en el tiempo durante un único espacio disponible:

1) Splice_Request con time( ) configurado, el servidor utiliza el puerto 2000.
2) Splice_Request con PriorSession, el servidor utiliza el puerto 2001.
3) Splice_Request con PriorSession, el servidor utiliza el puerto 2002.
4) Splice_Request con PriorSession, el servidor utiliza el puerto 2003.
5) Splice_Request con PriorSession, el servidor utiliza el puerto 2000.
6) Splice_Request con PriorSession, port_selection_descriptor port = 2000, el servidor utiliza el puerto 2000.
7) Splice_Request con PriorSession, port_selection_descriptor port = 2003, el servidor utiliza el puerto 2003.

Espacio disponible siguiente:

1) Splice_Request con time( ) configurado, el servidor utiliza el puerto 2000.

II.2 Ejemplo informativo 2

Utilizar el port_selection_descriptor( ) para configurar dinámicamente un puerto. El puerto de base se establece estáticamente en el mensaje Init_Request.

IP:Port de base = 192.168.134.9:3000

Número de puertos = 1

Los siguientes eventos tienen lugar en el tiempo durante un único espacio disponible:

1) Splice_Request con time( ) configurado, el servidor utiliza el puerto 3000.
2) Splice_Request con PriorSession, port_selection_descriptor IP = 192.168.134.9 puerto = 2010, el servidor configura y utiliza 192.168.134.9:2010.
3) Splice_Request con PriorSession, port_selection_descriptor IP = 239.192.0.2 puerto = 2010, el servidor configura y utiliza 239.192.0.2:2010.

Espacio disponible siguiente:

1) Splice_Request con time( ) configurado, el servidor utiliza el puerto 2000.

Hay que señalar que no se requiere el mismo número de puertos con IP de base, ya que cada espacio disponible puede asignar dinámicamente un número de puerto, según se ha descrito en el presente ejemplo.

BIBLIOGRAFÍA

Bibliografía

- KAR (M.), NARASIMHAN (S.), PRODAN (R.): Local Commercial Insertion in the Digital Headend, *Proceedings of NCTA 2000 Conference*, New Orleans, USA.
- Cable Television Laboratories: Cable Advertising, white paper, Louisville, CO, March 1997.

Dónde adquirir estas referencias

- The National Cable Television Association, 1724 Massachusetts Ave., NW, Washington, D.C. 20036-1969; Telephone: 202-775-3669; URL: http://www.ncta.com
- CableLabs, 400 Centennial Parkway, Louisville, CO 80027; Telephone: 303-661-9100; Facsimile: 303-661-9199; URL: http://www.cablelabs.com

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 multimedios
Serie I	Red digital de servicios integrados
Serie J	**Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedios**
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	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, 2006